T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

ANALYSIS AND DATA DISPLAY SYSTEM OF STOCK EXCHANGE DATA IN FREQUENCY DOMAIN

MELIHCAN ÇİLEK

SUPERVISOR
ASSOC. PROF. MEHMET GÖKTÜRK

GEBZE
2023

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

ANALYSIS AND DATA DISPLAY SYSTEM OF STOCK EXCHANGE DATA IN FREQUENCY DOMAIN

MELIHCAN ÇİLEK

SUPERVISOR
ASSOC. PROF. MEHMET GÖKTÜRK

2023
GEBZE

	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 22/06/2023 by the following jury.

JURY

Member (Supervisor)	:	Assoc. Prof. Mehmet Göktürk
Member	:	Assoc. Prof. Mehmet Göktürk
Member	:	Prof. Dr. Hasari ÇELEBİ

ABSTRACT

An integrated analysis, data display, and anomaly detection system for stock exchange data in the frequency domain is presented. The primary goal is to design a comprehensive system that utilizes Fourier transform for frequency analysis and localized frequency information extraction. By incorporating interactive displays, such as time-frequency plots, users can explore and analyze market trends, while the anomaly detection module with two different algorithms used identifies irregularities and outliers in the data. Experimental evaluation using real-world data from 'Borsa Istanbul' demonstrates the system's effectiveness and visualization. This research contributes to the field of financial analysis by providing a powerful tool for understanding and interpreting stock market dynamics from a frequency domain perspective, with the added capability of detecting anomalies.

Keywords: stock exchange data, frequency domain analysis, data display system, Fourier transform, wavelet transform, anomaly detection, visualization, financial analysis, decision-making, trading strategies

ACKNOWLEDGEMENT

I would like to express my gratitude to my thesis advisor, Assoc. Prof. Mehmet Göktürk, for his guidance, support, and encouragement throughout this research and implementation of Analysis and Data Display System of Stock Exchange Data in Frequency Domain project. His expertise and feedback have been instrumental in the completion of this thesis. Thank you for your patience, understanding, and support. This thesis would not have been possible without you.

Melihcan Çilek

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
$x(t)$	:	Continuous-time signal
π	:	Ratio of a circle's circumference to its diameter, approximately 3.14159
$X(\omega)$	:	The frequency-domain representation of the signal
ω	:	Representation of frequency
t	:	Representation of time domain
j	:	Representation of imaginary unit
$\mathcal{F}^{-1}$	:	Inverse Fourier Transform
a_0	:	Coefficient using while calculating Fourier Series
a_n	:	Coefficient using while calculating Fourier Series
b_n	:	Coefficient using while calculating Fourier Series
$\mathscr{L}$	:	Laplace Transform
$F(s)$	:	The System's Frequency Response in Laplace Transform

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

The realm of stock exchange data analysis and decision-making is a complex and ever-evolving domain. With the exponential growth of financial markets and the abundance of data available, there is an increasing demand for advanced tools capable of effectively analyzing and visualizing stock exchange data. Frequency domain analysis has emerged as a valuable approach for understanding the underlying patterns and dynamics inherent in financial time series.

This thesis presents a comprehensive system that integrates analysis, data display, and anomaly detection specifically tailored for stock exchange data in the frequency domain. The primary objective is to develop a robust tool that utilizes advanced techniques, including Fourier transform , to extract meaningful frequency components from raw data. By leveraging this transform, the system aims to uncover hidden cyclic patterns and trends within the stock market data.

Effective data visualization plays a crucial role in financial analysis. Visual representations of data enable market participants, traders, and researchers to grasp complex relationships and trends intuitively. Therefore, the developed system incorporates interactive displays, such as time-frequency plots, allowing users to explore and interpret the frequency domain characteristics of the stock exchange data. These visualizations provide a deeper understanding of market dynamics and aid in identifying potential trading opportunities.

Another important aspect addressed in this research is the detection of anomalies within the stock exchange data. Anomalies, which refer to significant deviations from expected patterns or outliers, can provide valuable insights into irregular market behavior or exceptional trading opportunities. The implemented anomaly detection module assists in identifying such occurrences and alerting users to potential anomalies, enhancing informed decision-making and risk management.

The effectiveness and reliability of the proposed system are evaluated through extensive experimentation using real-world stock exchange data. The evaluation includes comparisons with existing methods and analysis of the system's impact on enhancing trading strategies and decision-making processes.

By providing an integrated analysis, data display, and anomaly detection system, this research aims to contribute to the field of financial analysis, equipping market participants, traders, and researchers with a powerful tool for understanding and interpreting

stock market dynamics from a frequency domain perspective. Ultimately, the thesis strives to enhance the accuracy and efficiency of trading strategies, facilitate informed decision-making, and unlock valuable insights within the stock exchange data.

1.1. Frequency Domain

The frequency domain analysis is a fundamental aspect of this thesis, providing a mathematical representation of signals that unveils their frequency components and allows for a comprehensive understanding of their underlying characteristics. It offers a distinct perspective by examining signals based on their frequency content rather than their time-domain representation.
In this research, the frequency domain analysis is achieved through the utilization of mathematical transforms, namely the Fourier transform and the wavelet transform. The Fourier transform serves as a powerful tool for converting signals from the time domain to the frequency domain. It decomposes signals into a sum of sinusoidal components at different frequencies, providing insights into the energy distribution across the frequency spectrum. Mathematically, the Fourier transform of a continuous-time signal $x(t)$ is defined as:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega$$

We see that $X(\omega)$ determines the frequency-domain representation of the signal, the weight (relative importance) of the exponential function, ω denotes the frequency, t represents time domain and j is the imaginary unit.
That function $X(\omega)$ is called the **Fourier Transform** of $x(t)$ and will denoted by $X(\omega) = F(x(t))$. The function often written as:

$$x(t) \xrightarrow{\mathcal{F}} X(\omega)$$

The Fourier transform (the analysis equation) is given by,

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt$$

As there is a one to one correspondence between a signal and its Fourier transform we can also write:

$$x(t) \xleftarrow{\mathcal{F}^{-1}} X(\omega) \quad or \quad X(\omega) \xrightarrow{\mathcal{F}^{-1}} x(t)$$

This arrow diagram evidently refers to the **inverse Fourier Transform** $\mathcal{F}^{-1}$ or the synthesis equation.

1.1.1. Advantages of using Frequency Domain

One compelling motivation for employing a frequency-domain representation in problem analysis is the simplification of mathematical analysis. This is particularly valuable for systems governed by linear differential equations, which constitute a significant class of systems with widespread real-world applications. Transforming the system's description from the time domain to the frequency domain converts the differential equations into algebraic equations, considerably easing their solution.

Moreover, examining a system from a frequency perspective often provides an intuitive understanding of its qualitative behavior. A rich scientific terminology has emerged to characterize the behavior of physical systems in response to time-varying inputs. Terms such as bandwidth, frequency response, gain, phase shift, resonant frequencies, time constant, resonance width, damping factor, Q factor, harmonics, spectrum, power spectral density, eigenvalues, poles, and zeros are employed to describe and analyze the system's properties and characteristics.

A prominent domain where frequency-domain analysis offers superior insights compared to the time domain is music. The theory of musical instruments' operation and the musical notation used to record and discuss musical compositions are inherently based on the decomposition of complex sounds into their constituent frequencies. By examining the frequency components and their relationships, a deeper understanding of the intricate aspects of music is achieved.

In summary, the utilization of frequency-domain analysis in problem-solving presents significant advantages, including the simplification of mathematical analysis through the conversion of differential equations to algebraic equations. Moreover, the frequency domain provides an intuitive understanding of system behavior, facilitated by a well-established scientific vocabulary that captures essential characteristics. The domain-specific example of music further exemplifies the value of frequency-domain analysis in gaining insights into complex phenomena and expanding our comprehension of diverse fields.

1.1.2. Types of Frequency Domain

While "the" frequency domain is often referred to in singular terms, it encompasses various mathematical transforms utilized for analyzing time-domain functions. These frequency domain methods are tailored to specific applications and fields. The following are the most commonly used transforms and their associated domains of application:

1. Fourier Series
2. Fourier transform
3. Laplace transform
4. Z-transform
5. Wavelet transform

In a broader sense, the concept of a transform domain can be extended to any transform used for analysis. While these specific transforms capture different aspects of frequency, they are collectively referred to as frequency domain methods due to their ability to reveal frequency-related information.

1.1.2.1. Fourier Series

Fourier series is a mathematical technique employed to express periodic functions as an infinite sum of sinusoidal components. It provides a systematic approach to decompose a periodic function $f(x)$ into its fundamental frequency and its harmonics. The Fourier series representation of $f(x)$ is given by:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n cos(nx) + b_n sin(nx))$$

where a_0, a_n and b_n are the Fourier coefficients. These coefficients are calculated using these formulas:

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)cos(nx)dx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)sin(nx)dx$$

The Fourier series representation allows for the analysis of the frequency content and properties of periodic signals. By examining the coefficients a_n and b_n, one can determine the magnitudes and phase relationships of the sinusoidal components at each frequency n. The fundamental frequency corresponds to $n = 1$, and the subsequent harmonics are integer multiples of the fundamental frequency.

Through Fourier series analysis, it becomes possible to understand the dominant frequencies, harmonic structures, and phase characteristics of the periodic function.

This mathematical framework finds applications in signal processing, telecommunications, physics, and engineering fields, enabling the accurate representation, synthesis, and manipulation of periodic signals.

1.1.2.2. Fourier Transform

The Fourier transform is a fundamental mathematical operation used to analyze the frequency content of a function or signal by transforming it from the time domain to the frequency domain. It provides a valuable tool for understanding the underlying frequency components of a given function.
For a function $f(t)$ defined in the time donain, The Fourier Transform $F(\omega)$ is obtained by integrating $f(t)$ multiplied by a complex exponential function $e^{-i\omega t}$ over the entire real line:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-i\omega t} dt$$

Here, ω represents the frequency variable and i denotes the imaginary unit.

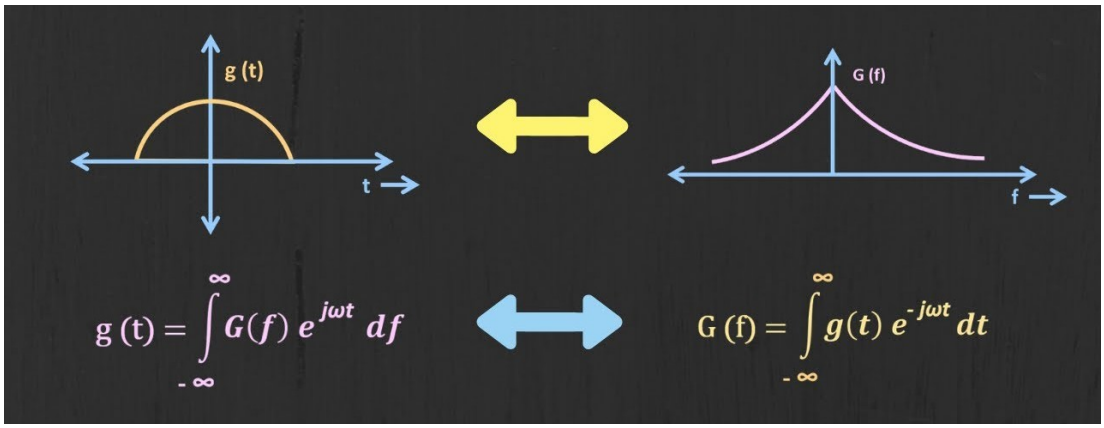

Figure 1.1: The Fourier Transform is applied to the image, revealing its frequency components in the frequency domain. The resulting spectrum showcases the distribution of frequencies present in the image, offering insights into patterns, structures, and textures that may not be apparent in the spatial domain. This analysis enables advanced image processing techniques.[1]

1.1.2.3. Laplace Transform

The Laplace transform is a mathematical technique that allows us to analyze and solve linear differential equations by transforming them from the time domain to the complex frequency domain. It provides a powerful tool for studying the behavior and

response of dynamic systems.
For function $f(t)$ defined for $t \geq 0$, the Laplace Transform $\mathscr{L}\{f(t)\}$ is defined as:

$$\mathscr{L}\{f(t)\} = F(s) = \int_0^\infty f(t) \cdot e^{-st} dt$$

Here, s is a complex variable representing the frequency parameter. The Laplace transform essentially captures the weighted sum of the function $f(t)$ with exponential decay factors e^{-st} for positive values if t.

The Laplace transform enables the analysis of differential equations by transforming them into algebraic equations in the frequency domain. This transformation simplifies the process of solving differential equations, as algebraic manipulations are often easier than solving differential equations directly.
In the frequency domain, the Laplace transform provides valuable insights into the system's response and stability. The transform of a function $f(t)$ yields the Laplace transform $F(s)$, which represents the system's frequency response. By analyzing the poles and zeros of the transformed function, we can determine stability, resonant frequencies, and other system properties.

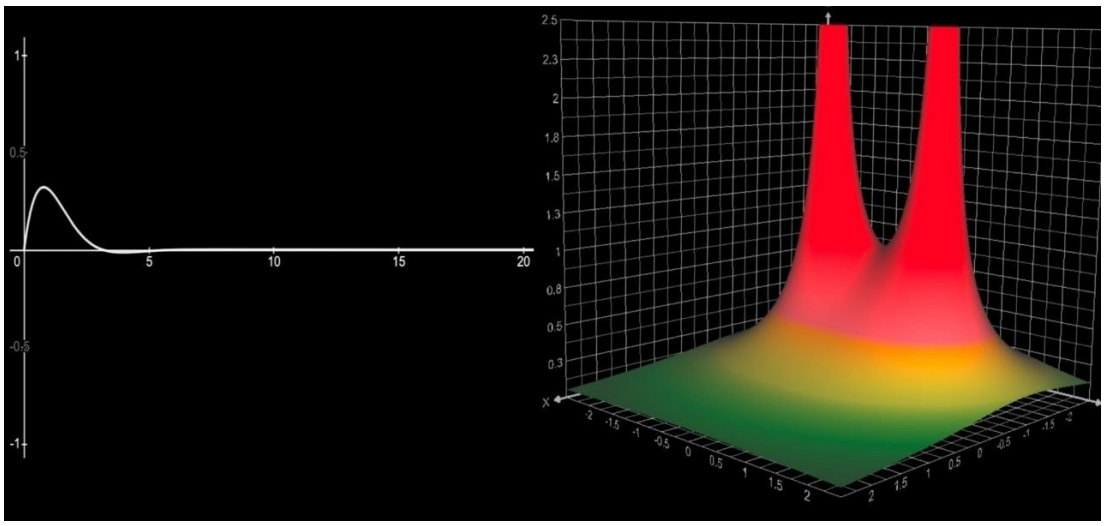

Figure 1.2: The Laplace Transform: A powerful mathematical tool used to analyze system behavior in the frequency domain, enabling the study of stability, response, and transient phenomena in various applications such as control systems and signal processing.[2]

1.1.2.4. Z-Transform

The Z-transform is a mathematical tool used to analyze discrete-time signals and systems in the frequency domain. It is analogous to the Laplace transform in the continuous-time domain. The Z-transform provides a convenient representation of discrete-time functions and allows for the analysis of their frequency response and system behavior.
For a discrete-time signal $x[n]$, the Z-transform $X(z)$ is defined as a power series in the complex variable z:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] \cdot z^{-n}$$

Here, z represents the complex frequency parameter, The Z-transform captures the weighted sum of the signal $x[n]$ with powers of z^{-1}, which correspond to different frequencies in the discrete-time domain.
The Z-transform allows us to analyze the frequency response of discrete-time systems. By applying the Z-transform to the difference equation that describes the system, we obtain a transfer function representation in the z domain. This transfer function provides insights into the system's frequency response, stability, and other characteristics.

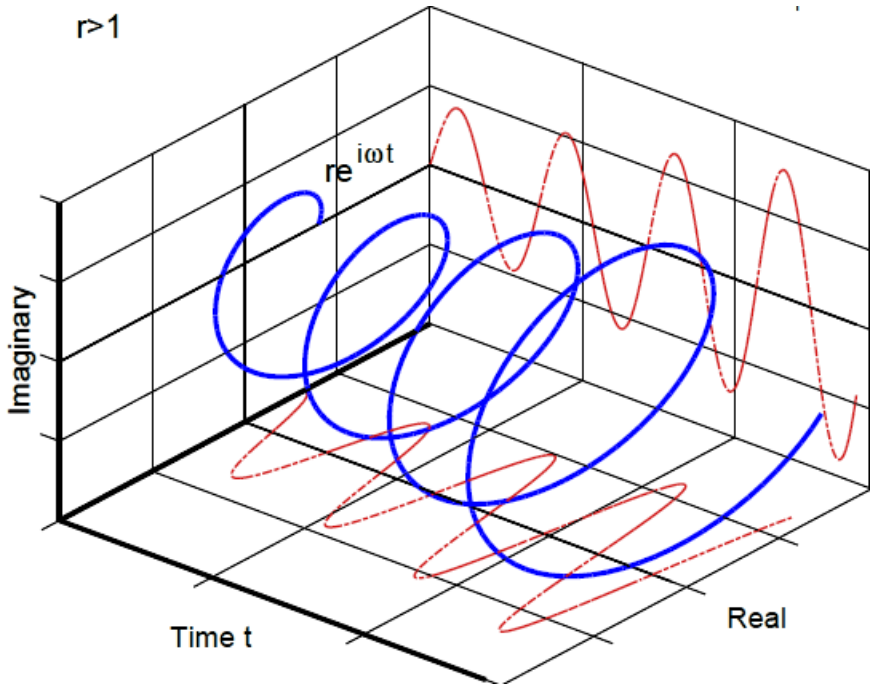

Figure 1.3: The Z-Transform: Analyzing discrete-time signals and systems in the frequency domain, providing insights into their characteristics and facilitating applications in digital signal processing and control systems.[3]

1.1.2.5. Wavelet Transform

The wavelet transform is a mathematical technique used for signal analysis that offers simultaneous representation of signals in both the time and frequency domains. It employs a set of basis functions called wavelets, which are localized in both time and frequency.

Given a signal $x(t)$ the continuous wavelet transform (CWT) is obtained by convolving $x(t)$ with scaled and translated versions of mother wavelet function $\psi(t)$:

$$CWT(a,b) = \int_{-\infty}^{\infty} x(t) \cdot \frac{1}{\sqrt{a}} \psi(\frac{t-b}{a}) dt$$

Here, a represents the scale parameter, controlling the width of the wavelet, and b represents the translation parameter, determining the position of the wavelet. The wavelet function $\psi(t)$ must satisfy certain admissibility conditions to ensure accurate representation and efficient analysis.

The discrete wavelet transform (DWT) is a computationally efficient version of the wavelet transform that operates on discrete signals. The DWT operates on discrete-time signals and provides a multiresolution representation. It involves a series of high-pass and low-pass filters followed by downsampling. Let $x[n]$ be a discrete-time signal, and consider a pair of filters: a high-pass filter $h[n]$ and a low-pass filter $g[n]$. The DWT coefficients at a certain level j are computed as:

$$c_j[n] = \sum_k x[k] \cdot h[n - 2^j k]$$

$$d_j[n] = \sum_k x[k] \cdot g[n - 2^j k]$$

where $c_j[n]$ represents the approximation coefficients and $d_j[n]$ represents the detail coefficients at level j. The downsampling operation reduces the sampling rate by a factor of 2.

The DWT recursively applies this filtering and downsampling process to the approximation coefficients at each level, resulting in a hierarchical representation of the signal. The approximation coefficients at each level capture the low-frequency components or coarse details, while the detail coefficients represent the high-frequency components or fine details of the signal.

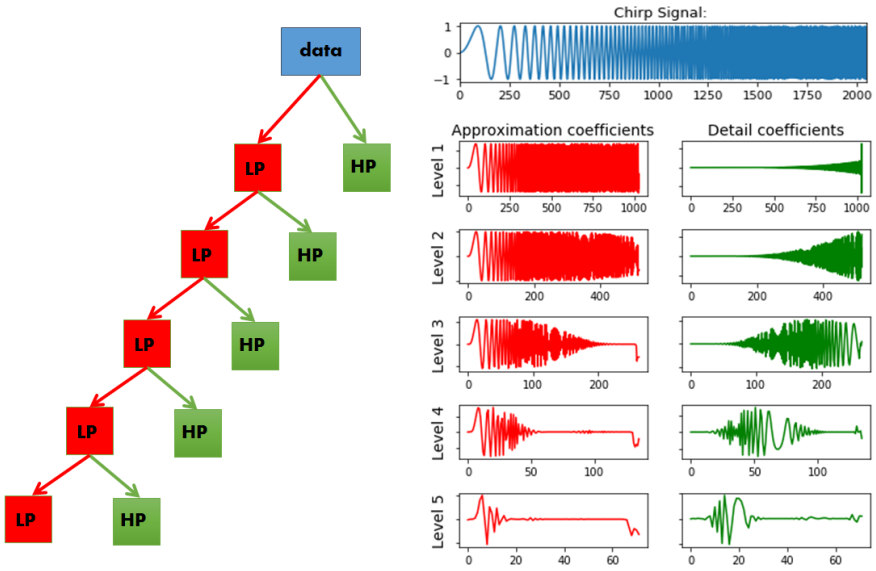

Figure 1.4: Wavelet Transform: A versatile mathematical tool used to analyze signals and images at multiple scales, capturing both local and global information. By using wave functions with different frequencies and durations, the Wavelet Transform reveals fine details and overall trends in the data. [4]

1.1.3. Discrete Frequency Domain

In the realm of frequency analysis, a discrete frequency domain refers to a frequency domain that is characterized by a discrete set of values rather than a continuous range. The discrete nature of this domain is exemplified by the discrete Fourier transform, which maps a function with a discrete time domain to a function with a corresponding discrete frequency domain. Conversely, the discrete-time Fourier transform maps functions with discrete time (i.e., discrete-time signals) to functions that possess a continuous frequency domain.

When considering the Fourier transform of a periodic signal, it exhibits energy solely at a fundamental frequency and its harmonics. In other words, a periodic signal can be effectively analyzed using a discrete frequency domain. Conversely, a discrete-time signal gives rise to a periodic frequency spectrum. By combining these two principles, when dealing with a time signal that is both discrete and periodic, the resulting frequency spectrum is also characterized by discreteness and periodicity. This scenario aligns with the common context for utilizing a discrete Fourier transform.

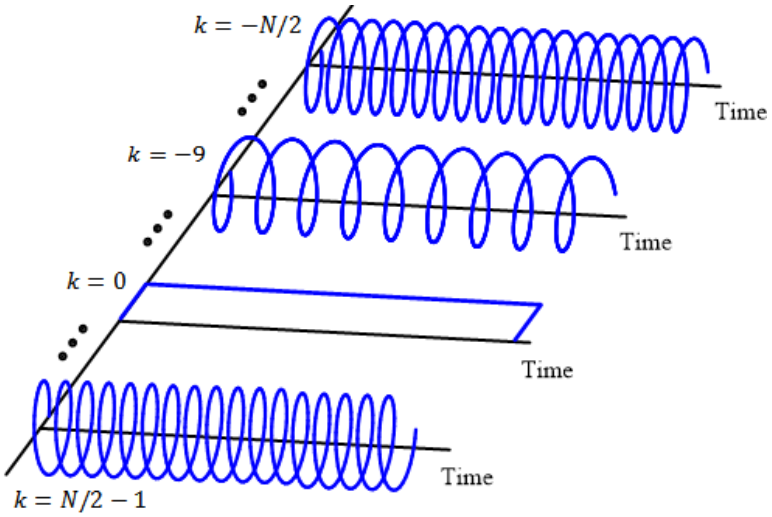

Figure 1.5: Discrete Frequency Domain: A representation of signals or data in discrete form, showcasing the distribution of frequencies present in the signal. By converting signals to the frequency domain, it enables analysis, filtering, and manipulation of specific frequency components. [5]

1.1.4. Fourier Transform Types

There are three main types of fourier transform, each one is suited for different scenarios. The main types of Fourier Transforms are as follows:

1. Continuos Fourier Transform (CFT): The continuous Fourier transform is defined for continuous-time signals. It transforms a continuous-time function from the time domain to the frequency domain. The formula for the CFT is given by:

$$F(\omega) = \int f(t) \cdot e^{-j\omega t} dt$$

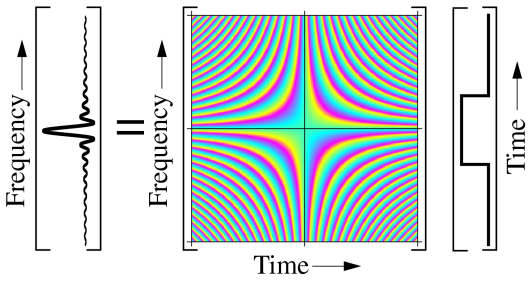

Figure 1.6: Continuous Fourier Transform: Analyzing continuous-time signals by decomposing them into their frequency components.[6]

2. Discrete Fourier Transform (DFT): The discrete Fourier transform is used to analyze discrete-time signals or sampled signals. It converts a sequence of N discrete samples from the time domain to the frequency domain. The DFT is defined by the equation:

$$F[n] = \sum_{k=0}^{N-1} f[k] \cdot e^{-j\frac{2\pi}{N}nk}$$

where

$$n = 0 \rightarrow N - 1$$

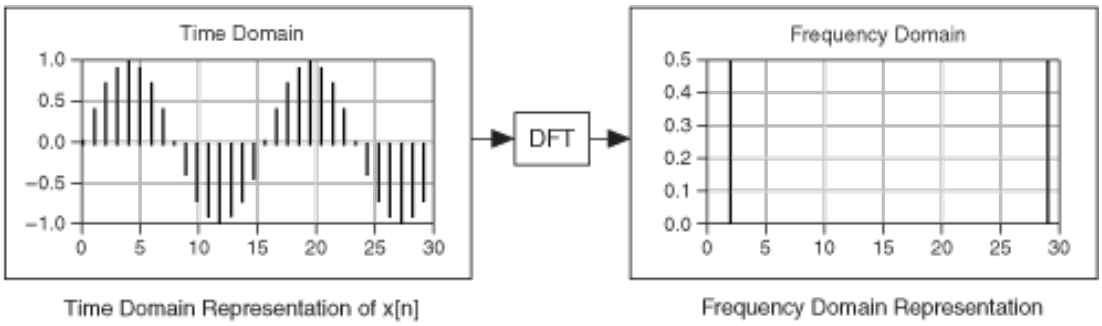

Figure 1.7: Discrete Fourier Transform: Analyzing discrete-time signals by decomposing them into frequency components, facilitating efficient analysis and manipulation in digital signal processing and image processing.[7]

3. Fast Fourier Transform (FFT): The fast Fourier transform is an efficient algorithm for computing the DFT. It reduces the computational complexity of the DFT from $O(N^2)$ to $O(NlogN)$, making it widely used in various applications. The FFT essentially performs the same mathematical operation as the DFT but in a more optimized manner.

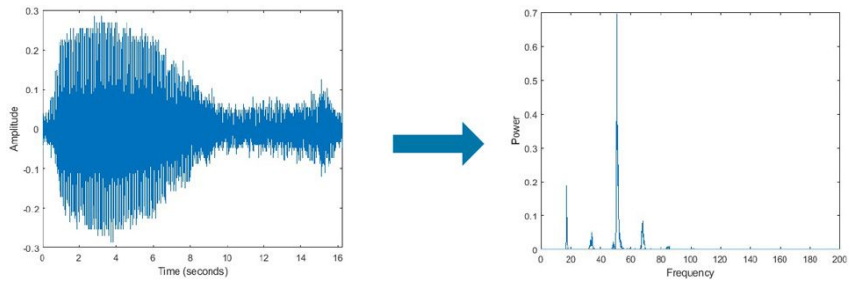

Figure 1.8: Fast Fourier Transform: A rapid computational algorithm for efficiently computing the Fourier Transform of a discrete-time signal or data. By converting the signal from the time domain to the frequency domain, FFT provides valuable insights into the signal's frequency content and enables efficient spectral analysis.[8]

1.1.5. Fast Fourier Transform and Why is it used?

The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform (DFT). It revolutionized the field of signal processing by significantly reducing the computational complexity of the DFT, making it practical for real-time and large-scale applications.
Here are a few key reasons why the FFT is widely used:

1. Computational Efficiency: The FFT algorithm reduces the complexity of computing the DFT from $O(N^2)$ to $O(NlogN)$, where N is the number of samples in the input sequence. This improvement in efficiency makes it feasible to perform Fourier analysis on large datasets and real-time signal processing tasks. The FFT enables rapid calculations of frequency spectra, convolution, correlation, and other operations involving Fourier transforms.

2. Real-Time Signal Processing: Many applications, such as audio and video processing, communication systems, and medical imaging, require real-time analysis of signals. The FFT's speed and efficiency allow for efficient processing of signals in real-time, enabling tasks like noise filtering, compression, modulation, and demodulation to be performed on-the-fly.

3. Spectrum Analysis: The FFT provides a powerful tool for analyzing the frequency content of a signal. By transforming a time-domain signal into the frequency domain, the FFT reveals the amplitudes and phases of various frequency components present in the signal. This information is crucial in fields such as audio processing, vibration analysis, image processing, and telecommunications, where understanding the frequency characteristics of a signal is essential.

4. Convolution and Filtering: The convolution operation, which combines two signals to produce a third signal, is fundamental in many applications, such as digital filtering and image processing. The FFT can efficiently compute convolutions by exploiting the convolution theorem, which states that convolution in the time domain is equivalent to multiplication in the frequency domain. This property allows for fast and accurate filtering of signals using frequency domain techniques.

5. Data Compression: The FFT plays a crucial role in various data compression techniques, including audio, image, and video compression algorithms. By transforming a signal into the frequency domain, it is possible to identify and exploit redundancies or perceptual limitations, resulting in more efficient encoding and compression of the data.

1.2. Anomaly and Its Detection

1.2.1. What is Anomaly?

An anomaly refers to something that deviates from what is expected or considered normal within a given context or system. It represents an observation, event, or data point that stands out from the usual patterns or behaviors. Anomalies can occur in various domains, including statistics, data analysis and security.

In statistics and data analysis, anomalies are often referred to as outliers. They are data points that significantly differ from the majority of the data points in a dataset, either by being exceptionally high or low. Identifying and understanding anomalies can be valuable in various applications, such as detecting fraudulent activities, monitoring system performance, or identifying unusual patterns in data.
In security, anomalies can indicate potential threats or malicious activities. Anomaly detection techniques are employed to identify abnormal network behavior, unauthorized access attempts, or anomalous patterns in user behavior, helping to detect security breaches and prevent cyber attacks.

1.2.1.1. Anomaly in Graphs

Anomalies in graphs refer to atypical or unexpected behavior observed within the context of a graph or dataset. They represent data points, patterns, or structures that significantly deviate from the established norms or anticipated patterns. Anomalies may manifest as outliers, abrupt and extreme fluctuations, discontinuities, or other irregularities that do not conform to the overall trends or patterns exhibited by the majority of the data.

The detection and analysis of anomalies in graphs play a crucial role in various domains, including but not limited to network traffic analysis, fraud detection, system monitoring, and social network analysis. The identification of anomalies allows for the detection of unusual events, outliers, or potential anomalies that necessitate further investigation or remedial action.

Graph anomalies can arise from various sources, including measurement errors, data corruption, system faults, or the presence of significant events or anomalies in the underlying processes being represented by the graph. Their detection requires the application of sophisticated analytical techniques that involve statistical methods, machine learning algorithms, graph theory, or pattern recognition approaches.

By effectively identifying and analyzing anomalies in graphs, it becomes possible to gain valuable insights into exceptional events, rare occurrences, or data points that may have a significant impact on the interpretation, decision-making, or risk assessment within the specific domain of application.

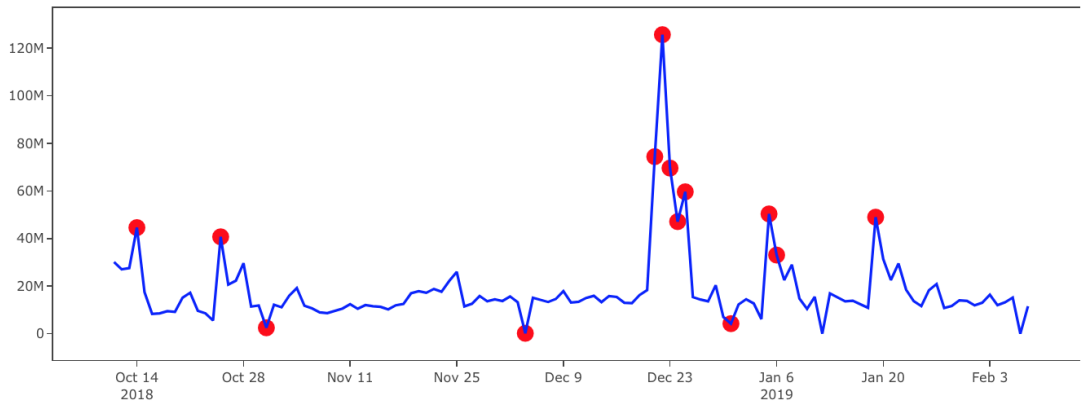

Figure 1.9: Anomalies in Graphs: Unveiling atypical patterns and outliers that deviate significantly from expected behavior within graph structures, leading to valuable insights for decision-making and analysis in diverse domains.[8]

1.2.2. Why do we detect anomalies?

Anomaly Detection (Outlier Detection) [9] in graphs refers to the process of identifying and flagging atypical patterns, outliers, or irregularities within graph structures or datasets. It involves analyzing the data to uncover deviations from the expected or normal behavior, enabling the identification of exceptional events, errors, or potential risks. Detecting anomalies in graphs plays a crucial role in various domains, including system monitoring, fraud detection, network analysis, and data quality assurance, as it allows for proactive measures, decision-making based on reliable information, and the optimization of processes.

The primary reason for detecting anomalies in graphs is the identification of unusual events. Anomalies represent unexpected occurrences that deviate significantly from the regular patterns within the graph. By detecting these anomalies, we can identify and investigate occurrences that require immediate attention, such as system failures, security breaches, or fraudulent activities. Detecting and understanding these unusual events allows us to take appropriate actions, mitigate risks, and maintain the integrity of the system or process being analyzed.

Furthermore, anomaly detection in graphs serves as a mechanism for error detection and data quality assurance. Anomalies can arise from errors in data collection, data entry, or data transmission, leading to inaccuracies or inconsistencies in the graph. By detecting and flagging these anomalies, we can identify potential data quality issues and ensure the accuracy and reliability of the underlying data. This step is essential for maintaining data integrity and ensuring that subsequent analysis or decision-making processes are based on trustworthy information.

Anomaly detection also plays a crucial role in performance monitoring and fault diagnosis. By continuously monitoring and detecting anomalies in real-time, we can identify performance bottlenecks, system failures, or irregular behavior in various processes. Timely detection of anomalies aids in diagnosing and resolving issues promptly, minimizing downtime, and ensuring optimal system performance.

Moreover, anomaly detection in graphs is instrumental in detecting fraud and security threats. Anomalies can reveal patterns or behaviors indicative of fraudulent activities or security breaches. By analyzing the graph data and detecting anomalous patterns, we can identify potential instances of fraud, unauthorized access, or suspicious behavior. Anomaly detection in graphs helps in detecting anomalies in transactions, network traffic, user behaviors, or system logs, facilitating proactive measures to prevent and mitigate security threats.

Lastly, the detection of anomalies in graphs provides valuable insights for decision-making and optimization. Anomalies often contain hidden patterns or trends that are not apparent in regular data. By detecting and analyzing these anomalies, we can gain a deeper understanding of underlying processes, uncover emerging trends, identify potential opportunities, or assess risks. Anomaly detection aids in making data-driven decisions, developing effective strategies, and optimizing performance in various domains, including finance, marketing, operations, and risk management.

1.2.3. Anomaly Detection Techniques

Anomaly detection techniques encompass a range of methodologies and algorithms employed to identify and flag atypical patterns, outliers, or irregularities within datasets. These techniques aim to distinguish between normal and abnormal behaviors to detect and investigate unusual events or anomalies that significantly deviate from expected patterns. Some commonly used anomaly detection techniques include:

1. Statistical Methods: Statistical methods play a crucial role in anomaly detection by utilizing various statistical measures and techniques to identify anomalies

based on deviations from expected patterns. Here are some commonly used statistical techniques in anomaly detection:

(a) Z-Score: The z-score technique measures how many standard deviations a data point is away from the mean. Data points with z-scores above a certain threshold are considered anomalies. This method assumes that the data follows a normal distribution.[10]

(b) Percentile-Based Approaches: Percentile-based techniques involve defining thresholds based on percentiles of the data distribution. Data points that exceed a specified percentile (e.g., 95th percentile) are identified as anomalies. This method is particularly useful when the data does not follow a normal distribution.[11]

(c) Interquartile Range (IQR): The IQR is a measure of statistical dispersion that quantifies the spread of data within the first and third quartiles of a dataset. The IQR technique identifies outliers by considering data points that fall outside a specified range, typically defined as the first quartile minus a multiplier times the IQR and the third quartile plus the same multiplier times the IQR. [12]

(d) Box Plot: A box plot visualizes the distribution of data through quartiles. It identifies outliers as data points that fall below the first quartile minus a specified factor (lower whisker) or above the third quartile plus the same factor (upper whisker). Data points beyond these whiskers are considered anomalies. [13]

2. Machine Learning Algorithms: Machine learning algorithms play a crucial role in anomaly detection by leveraging patterns and relationships in data to identify anomalous instances. Here is an explanation of machine learning algorithms commonly used in anomaly detection:

(a) Supervised Learning Algorithms: Supervised learning algorithms require labeled data, where anomalies are explicitly marked, to train a model. The algorithm learns the patterns and characteristics of normal instances and uses them to classify new data points as normal or anomalous. Examples of supervised learning algorithms used in anomaly detection include Support Vector Machines (SVM), Decision Trees, and Random Forests. [14]

(b) Unsupervised Learning Algorithms: Unsupervised learning algorithms work with unlabeled data, making them suitable for unsupervised anomaly detection. These algorithms aim to discover patterns, structures, or outliers that deviate from the majority of the data. Clustering algorithms such

as k-means, Gaussian Mixture Models (GMM), and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) are commonly used in unsupervised anomaly detection. [15]

(c) Semi-Supervised Learning Algorithms: Semi-supervised learning algorithms combine labeled and unlabeled data. They initially learn patterns from the labeled data, representing normal behavior, and then use the unlabeled data to identify instances that deviate significantly from the learned patterns as anomalies. This approach is useful when labeled anomaly data is scarce or expensive to obtain. [16]

(d) Deep Learning Algorithms: Deep learning algorithms, such as Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs), excel at capturing complex patterns and representations in large-scale datasets. They can automatically learn hierarchical representations of data, making them well-suited for anomaly detection in various domains, including images, text, and time series data. [17]

3. Time Series Analysis: Time series analysis techniques are specifically designed to detect anomalies in temporal data. These methods consider the temporal dependencies and patterns in the data to identify deviations from expected behavior. Techniques like autoregressive integrated moving average (ARIMA) models, exponential smoothing, and change point detection are commonly used for anomaly detection in time series data. Here is an explanation of these techniques in the context of anomaly detection:

   (a) Autoregressive Integrated Moving Average (ARIMA) Models: ARIMA models are used to capture the temporal dependencies and trends in time series data. They incorporate the concepts of autoregression (AR), differencing (I), and moving average (MA) to model and predict future values. In anomaly detection, ARIMA models can be trained on historical data to forecast expected values and compare them with the actual observations. Anomalies are identified when there are significant deviations between the predicted and observed values. [18]

   (b) Exponential Smoothing: Exponential smoothing is a technique that assigns exponentially decreasing weights to past observations in a time series. It is useful for detecting anomalies by smoothing out the noise and capturing the underlying trend and seasonality in the data. Exponential smoothing methods, such as simple exponential smoothing or Holt-Winters' triple exponential smoothing, can be employed to generate smoothed forecasts and identify anomalies based on the differences

between the actual and predicted values. [19]

(c) Change Point Detection: Change point detection algorithms are designed to identify abrupt changes or shifts in the underlying behavior of a time series. These techniques are useful for detecting anomalies when there are significant departures from the expected pattern. Change points can indicate anomalous events, regime shifts, or sudden shifts in the statistical properties of the data. Various change point detection algorithms, such as the CUSUM (cumulative sum) algorithm or the Bayesian change point detection method, can be applied to identify anomalies in time series data. [20]

4. Nearest Neighbor Approaches: Nearest neighbor-based techniques assess the similarity between data points to detect anomalies. These methods calculate distances or similarities between data points and identify instances that are significantly different from their neighbors. Examples include k-nearest neighbor (k-NN) and Local Outlier Factor (LOF) algorithms. Here's an explanation of these techniques:

   (a) K-Nearest Neighbor (k-NN) Algorithm: The k-NN algorithm classifies data points based on their proximity to other points in a feature space. In the context of anomaly detection, the k-NN algorithm is used to determine the distance between a data point and its k nearest neighbors. The idea is that normal instances are surrounded by similar instances, while anomalies are often isolated or differ significantly from their neighbors. By setting a threshold distance, the algorithm can identify instances that are farther away from their neighbors as potential anomalies. [21]

   (b) Local Outlier Factor (LOF) Algorithm: The LOF algorithm measures the local density deviation of a data point compared to its neighbors. It determines the degree of anomalousness by considering how isolated or dissimilar a point is in its local neighborhood. The LOF algorithm assigns an anomaly score to each instance based on its density compared to the density of its neighbors. [22]

1.3. Contribution

In this thesis, the focus is on anomaly detection in stock exchange data using frequency domain analysis techniques, specifically the Interquartile Range (IQR) algorithm and wavelet transformation.
The IQR algorithm is employed as a statistical method for anomaly detection in the

frequency domain. By calculating the IQR values for the frequency components of the data, the system identifies outliers and abnormal behaviors. This approach helps in detecting anomalies that deviate significantly from the expected distribution of frequencies.

Wavelet transformation, on the other hand, provides a multiresolution analysis of the stock exchange data. It allows for the identification of anomalies at different scales or resolutions, making it suitable for detecting transient or localized behaviors in the frequency domain. This technique enhances the system's ability to capture various types of anomalies.

By incorporating both the IQR algorithm and wavelet transformation, the analysis and data display system in this thesis offer a comprehensive approach to anomaly detection. The system compares the results obtained from these two techniques, providing a more robust assessment of the detected anomalies.

Visualizations are utilized to display the frequency domain analysis results and the detected anomalies. Graphs and charts present the frequency spectrum with highlighted anomalies identified by each method. These visual representations assist in the interpretation and understanding of anomalies, aiding in the decision-making process.

Overall, the combination of the IQR algorithm and wavelet transformation in the anomaly detection system provides a comprehensive framework for identifying and analyzing anomalies in stock exchange data. This research contributes to the field of anomaly detection by offering insights into the effectiveness of these techniques in the frequency domain.

2. ANALYSIS AND DATA DISPLAY SYSTEM OF STOCK EXCHANGE DATA IN FREQUENCY DOMAIN

2.1. Process of obtaining data from Borsa Istanbul

In order to conduct the data analysis in this study, it was essential to acquire the necessary data from Borsa İstanbul. The data acquisition process involved the utilization of an ”Academic Application” created specifically for this purpose, accessed through the official website of Borsa İstanbul at Borsa Istanbul Datastore.

By logging in via the Academic Application, we gained authorized access to the relevant data source, namely the ”Equity Market - Intraday Transaction Book.” This information-rich dataset was obtained from Borsa İstanbul through a meticulous and extensive transaction process, ensuring the integrity and accuracy of the data used in our analysis.

Throughout this endeavor, we would like to extend our heartfelt gratitude to the officials at Borsa İstanbul who provided invaluable assistance and support. Their cooperation and guidance greatly facilitated the successful acquisition of the necessary data, enabling us to conduct in-depth analysis and make meaningful interpretations.

We acknowledge the Borsa İstanbul officials for their responsiveness, expertise, and professionalism in helping us navigate through the data acquisition process. Their willingness to collaborate and share their knowledge played a significant role in the successful completion of this study.

The cooperation between the research team and Borsa İstanbul officials exemplifies the importance of industry-academia partnerships in advancing scientific knowledge and fostering practical applications. The collaboration between academia and Borsa İstanbul not only contributed to the research conducted in this study but also laid the foundation for potential future collaborations and mutual benefits.

In light of the above, we express our sincere appreciation to Borsa İstanbul for their support and cooperation throughout the data acquisition process. Their contribution has been instrumental in the successful execution of this study, and we are grateful for the opportunity to work closely with such a respected and esteemed institution.

2.2. Parse The Data

Once the data was obtained, a meticulous examination of its structure was carried out to understand the composition of columns and rows. This analysis was crucial in determining the specific information required and devising an effective data retrieval strategy. Given the substantial size of the dataset, various data manipulation techniques were explored to handle the data efficiently and ensure its usability for further analysis.

To streamline the data manipulation process, a comparative evaluation of different libraries was conducted. This involved assessing the capabilities and functionalities of popular data manipulation libraries, such as Pandas, NumPy, and SQL, among others. By comparing these libraries, we aimed to identify the most suitable tool for handling and transforming the acquired dataset.

Data manipulation techniques were employed to perform a range of operations, including data cleaning, preprocessing, aggregation, filtering, and transformation. Missing values were addressed through imputation or removal, ensuring data completeness. Inconsistent or erroneous data entries were corrected or flagged for further investigation.

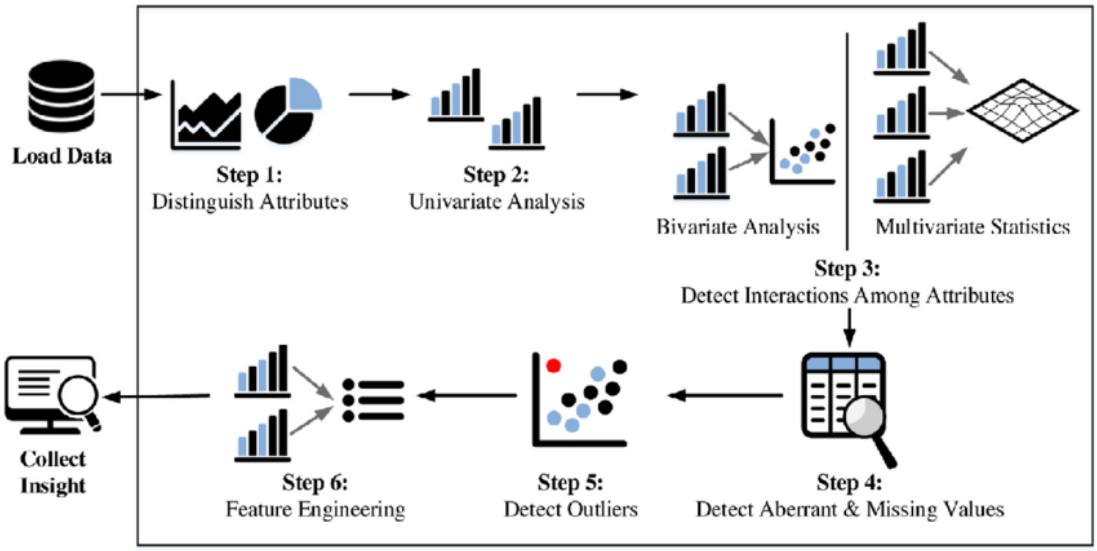

Figure 2.1: Steps of Data Analysis: A systematic process for extracting insights from data, including data acquisition, preprocessing, exploratory data analysis, feature engineering, statistical analysis, machine learning modeling, model evaluation, interpretation and visualization, and conclusion and reporting.[23]

Furthermore, feature engineering techniques were applied to extract meaningful insights from the raw data. This involved creating new variables or transforming existing ones to capture relevant patterns and relationships. Techniques like scaling, normalization, and encoding were utilized to ensure compatibility and consistency in the data.

Additionally, exploratory data analysis techniques were employed to gain a deeper understanding of the dataset's characteristics and uncover potential patterns or anomalies. Descriptive statistics, visualizations, and statistical tests were utilized to reveal key insights and support decision-making processes.

The selection of appropriate data manipulation techniques and libraries played a crucial role in effectively managing and preparing the dataset for analysis. By leveraging the strengths of specific libraries and employing various data manipulation techniques, we aimed to optimize the efficiency and accuracy of subsequent analyses and facilitate the interpretation of results.

2.3. Reconstructing The Parsed Data in a Different Form

The incoming dataset, consisting of various stock trading variables on Borsa Istanbul, underwent a series of data analysis procedures to extract meaningful insights. To streamline the analysis process, several data analysis terminologies and techniques were applied.

One important step in the data analysis pipeline was feature selection, where irrelevant or redundant columns were removed from the dataset. This process involved identifying variables that did not contribute significantly to the analysis objectives, such as transaction type, activity, and price, and excluding them from further consideration. By focusing on the most relevant features, the analysis could be performed more efficiently and accurately.

The data was then processed using a chunking technique, which involves dividing the dataset into smaller, manageable portions or chunks. This approach enables the analysis of large datasets that may not fit entirely into memory. By sequentially loading and processing these chunks, computational resources are utilized more effectively, ensuring efficient data processing.

Grouping the data according to seconds provided a temporal dimension for analysis. This allowed for the identification of anomalies or irregularities that occurred within specific time intervals. By aggregating data at the second level, patterns and trends

in stock trading activities could be examined more closely, enabling the detection of potential outliers or unexpected behavior.

An important aspect of the analysis involved calculating the total trading volume. This metric provides insights into the overall level of trading activity within each time interval. By analyzing the volume data, unusual spikes or dips in trading volume could be identified, which could indicate abnormal market behavior or exceptional trading events.

Throughout the analysis, various statistical measures and techniques were employed. Descriptive statistics, such as mean, median, standard deviation, and percentiles, were used to summarize and characterize the data distribution. Statistical tests, such as hypothesis testing or t-tests, were applied to assess the significance of observed differences or patterns in the data.

By leveraging these data analysis terminologies and techniques, the dataset was effectively processed, anomalies were detected, and insights were derived. These methods contributed to a thorough examination of stock trading data in the frequency domain, providing valuable information for decision-making and market analysis purposes.

Borsa Istanbul operates according to a specific schedule and remains closed on certain occasions, special days, and holidays. Recognizing the importance of accounting for these non-trading days in the data analysis process, a diligent effort was made to incorporate this information into the analysis pipeline.

To ensure the accuracy and reliability of the analysis, a comprehensive list of non-trading dates was obtained directly from Borsa Istanbul's official website (Borsa Istanbul Official Holiday List). This list encompassed holidays, public observances, and other specific days when the stock exchange was not operational. By obtaining this data, it became possible to identify and filter out the corresponding dates from the dataset.

By integrating this additional study of filtering non-trading dates, the analysis focused solely on the trading days, eliminating any potential distortions or biases introduced by non-trading periods. This approach ensured that the analysis accurately reflected the trading dynamics and patterns within the market, providing a more robust foundation for drawing meaningful insights and making informed decisions.

The careful consideration of non-trading dates demonstrates a meticulous approach to data analysis, taking into account the unique characteristics of the stock exchange and the impact of non-trading periods on the dataset. By effectively filtering out these dates, the analysis became more focused and aligned with the actual trading activities, enhancing the accuracy and reliability of the results.

This additional study not only highlights the attention to detail in the analysis process but also underscores the commitment to maintaining data integrity and ensuring that the analysis accurately represents the dynamics of the stock exchange on trading days.

2.4. Process to Extract Meaning

After the necessary data transformations were performed, the data had to be processed to extract meaningful insights. Recognizing the significance of frequency domain analysis in this context, extensive research was conducted to explore techniques and methodologies for transitioning from the time domain to the frequency domain.

The investigation focused on understanding the principles and concepts of frequency domain analysis, including the mathematical foundations and algorithms involved. Different approaches were explored to effectively convert the data from its original time-based representation into the frequency domain.

The research encompassed exploring various techniques such as the Fast Fourier Transform (FFT), which was widely used for efficient frequency domain analysis. The algorithms and implementations of the FFT were carefully studied to identify the most suitable approach for analyzing stock exchange data.

Considerable attention was given to selecting appropriate algorithms and methods that would best suit the specific requirements of the analysis. These algorithms were chosen based on their ability to accurately capture and represent the frequency components present in the data, enabling a comprehensive understanding of the underlying patterns and dynamics.

The research conducted on frequency domain analysis provided valuable insights into the application of these techniques to the stock exchange data. Informed decisions were made regarding the algorithms and methods to be implemented in the analysis pipeline based on the knowledge gained from this research. This ensured that the data was processed effectively, enabling a deeper understanding of the frequency characteristics and patterns within the stock exchange data.

The research on frequency domain analysis and the subsequent determination of suitable algorithms exemplified a rigorous and informed approach to data processing. By leveraging the advancements in frequency domain techniques, the analysis aimed to uncover hidden insights and valuable information within the stock exchange data.

2.5. Transition from Time Domain To Frequency Domain

During the exploration of techniques for transitioning from the time domain to the frequency domain, it was discovered that the Fourier Transform is one of the most widely used methods for this purpose. Given the nature of our data set, which required a thorough analysis of frequency characteristics, the Fourier Transform was deemed suitable for extracting valuable insights.

2.5.1. NumPy versus SciPy

To implement this transformation, extensive research was conducted to identify the most appropriate libraries that could effectively perform the Fourier Transform. The two popular Python libraries, ”SciPy” and ”NumPy,” emerged as prominent candidates known for their capabilities in signal processing and scientific computing.

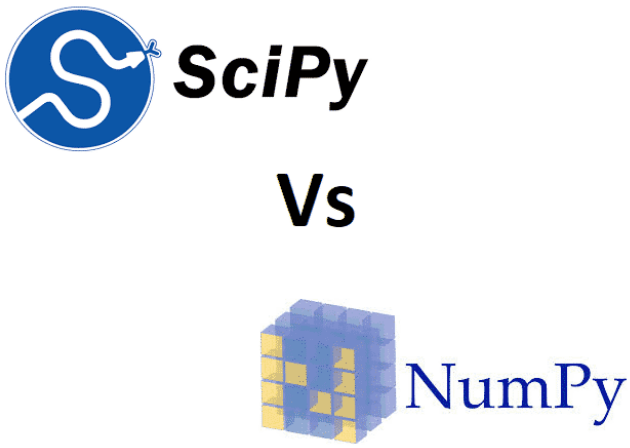

Figure 2.2

The decision-making process involved carefully evaluating the strengths and weaknesses of each library. While ”SciPy” offered a comprehensive suite of functions for scientific analysis, it was observed to be relatively slower and less efficient specifically for Fast Fourier Transform (FFT) operations. On the other hand, ”NumPy” provided a more focused and optimized set of functions for efficient FFT computations, aligning well with our requirement for speed and accuracy in analyzing the frequency domain data.

After careful consideration, the Fast Fourier Transform functions from the ”NumPy”

library were chosen for our analysis pipeline. This choice was motivated by the need for fast and efficient processing of the frequency domain data, ensuring timely and accurate results.

By leveraging the appropriate functions from the ”NumPy” library, we were able to seamlessly apply the Fourier Transform to the stock exchange data set. This enabled a comprehensive understanding of the frequency components and patterns embedded within the data, providing valuable insights into market dynamics and behaviors.

The utilization of the ”NumPy” library for the Fast Fourier Transform underscored a deliberate and well-informed decision, considering both the computational efficiency and effectiveness in analyzing the frequency domain. This approach ensured that the analysis was performed in a timely manner, allowing for more informed decision-making processes based on the frequency characteristics of the stock exchange data.

2.5.2. Pandas

For a brief explanation why Pandas used is that Pandas is a powerful and widely used open-source data manipulation and analysis library in Python. It offers a wide range of features and functionalities that make it well-suited for handling and processing data in various domains, including stock exchange data analysis.

2.6. Discovery of the Wavelet Transform - A Different Way

During the search for articles on the Fast Fourier Transform, the researcher came across an intriguing Turkish article titled ”Frequency Domain Approach To Financial Time Series Analysis: Wavelets Methods and An Evidence From IMKB” [24]. This article caught their attention and prompted them to delve deeper into the topic of Wavelet Transform. Intrigued by its potential applications, the researcher embarked on a research journey to explore the relevance and suitability of Wavelet Transform in their own project.

Encountering this article provided a valuable opportunity for the researcher to expand their understanding of frequency domain analysis techniques beyond the conventional Fast Fourier Transform. Recognizing the potential benefits of Wavelet Transform in financial time series analysis, the researcher conducted further research to explore its practical implications and its alignment with their project requirements.

The researcher’s objective was to incorporate the principles and methodologies of Wavelet Transform as an additional option in their analysis pipeline. This involved

gaining a thorough understanding of the underlying theory, considering practical implementation considerations, and evaluating the potential advantages of Wavelet Transform in capturing localized frequency information within financial time series data.

Through this exploration, the researcher aimed to broaden the scope of their project by considering alternative frequency domain analysis techniques. This approach would enable a more comprehensive and robust analysis of the stock exchange data they were working with. The research journey exemplified the researcher's commitment to an iterative scientific exploration process, as they remained open to adapting and incorporating new approaches based on emerging evidence and insights from the literature.

2.6.1. PyWavelets - Wavelet Transforms

PyWavelets is a Python library that provides functionality for performing wavelet analysis, including wavelet transforms, wavelet packet decompositions, and various wavelet-based signal processing techniques.

1. Wavelet Analysis: Perform wavelet transforms and decompose signals into different frequency components.
2. Multi-Resolution Analysis: Decompose signals into different levels or scales for hierarchical representation.
3. Visualization: Visualize wavelet coefficients, reconstructed signals, and decomposition trees for better understanding of the data.

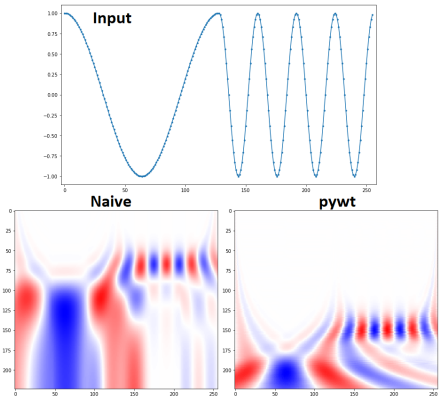

Figure 2.3: This example shows what is the difference between native way to implement wavelets and usage of PyWavelets [25]

2.7. Creation of Necessary Components for UI

With the converted data at hand, the focus shifted towards developing a user-friendly interface that would allow users to interact with and select various options for data display. This crucial step involved extensive research to design an intuitive and efficient user interface. The goal was to identify libraries and frameworks that provided the necessary components and functionalities for creating an interactive interface.

The research process involved exploring different options, considering factors such as ease of use, flexibility, and compatibility with the project requirements. Several libraries were assessed, each offering unique features and capabilities. By evaluating their documentation, community support, and compatibility with the chosen programming language, the most suitable libraries were identified.

Once the appropriate libraries were selected, the implementation of the user interface components began. This involved designing and integrating interactive elements such as dropdown menus, checkboxes, radio buttons, sliders, and buttons. These components were carefully chosen to provide users with a seamless and intuitive experience when selecting their desired data display options.

Furthermore, attention was given to the aesthetics and visual presentation of the data. The user interface was designed to ensure clear and visually appealing data visualization, enabling users to easily interpret and analyze the displayed information. Customization options, such as color schemes and plot styles, were also incorporated to enhance the user experience and accommodate individual preferences.

By integrating the user interface components with the data processing techniques, users could effortlessly select their desired data display options and observe the results in real-time. This interactive approach facilitated a dynamic and customizable experience, empowering users to gain insights from the stock exchange data in a way that best suited their needs.

2.7.1. Tkinter

After thorough research and evaluation of different user interface libraries, it was determined that the ”tkinter” toolkit was a comprehensive and versatile package that met the requirements for creating the desired user interface. The decision to utilize the ”tkinter” toolkit was based on its robust functionality, extensive documentation, and wide community support.

The ”tkinter” toolkit, which is included as a standard library in Python, provides a rich set of features and components for building graphical user interfaces. Its flexibility and ease of use make it a popular choice for creating interactive applications. With its extensive collection of widgets, including buttons, labels, entry fields, and canvas, ”tkinter” offered the necessary building blocks to design an intuitive and user-friendly interface for the analysis and data display system.

2.7.2. TkCalendar

Tkinter is a popular GUI toolkit in Python, but it doesn't provide a built-in calendar widget. TkCalendar, a third-party library built on top of Tkinter. It provides:

1. Calendar Functionality: TkCalendar provides a fully functional calendar widget that allows users to select dates, navigate through months, and view different years.
2. User-Friendly Interface: It offers an intuitive and visually appealing interface for displaying and interacting with calendar data.
3. Date Selection: TkCalendar simplifies the process of selecting dates by providing a convenient and interactive calendar interface.
4. Customization Options: TkCalendar allows customization of appearance and behavior, enabling developers to adapt the calendar widget to match the project's design and requirements. These kind of customizations also done in this project.

2.7.3. Matplotlib

Several packages were considered for creating the graphical interface, including ”Pyvis”, ”Plotly”, and ”Matplotlib”. After thorough research and testing, ”Matplotlib” was determined to be the most stable and efficient package, making it the preferred choice for creating the graphical interfaces in the project. Here are some of the reasons that selecting Matplotlib over other graph plotting libraries:

1. Stability: ”Matplotlib” is known for its stability and reliability. It has been extensively used in the scientific and data visualization communities for many years, making it a trusted choice for creating robust graphical interfaces.
2. Matplotlib Integration: ”Matplotlib” seamlessly integrates with other popular libraries in the Python ecosystem, such as NumPy and Pandas.
3. Cross-platform Compatibility: ”Matplotlib” is compatible with multiple operating systems, including Windows, macOS, and Linux.

2.7.4. mplcursors

The ”mplcursors” library is utilized in the project to facilitate the placement of annotations on the plotted graph. This library offers a convenient way to interactively annotate data points on the graph by displaying custom tooltips or labels near the cursor position. By using ”mplcursors,” users can easily add informative annotations to specific data points of interest, providing additional context and aiding in data analysis and interpretation. This feature enhances the overall visual presentation and makes it simpler to communicate key findings or insights derived from the plotted graph. Because ”mplcursors” library is compatible with matplotlib, the library used to leave highlighted annotations over graph plotted.

2.8. Display of Transformed Data in Graph

After executing the sequential operations, a graph is generated using the functions, if desired by the user. This marks the beginning of the analysis phase, where understanding the meaning conveyed by the graph becomes crucial. It is important to comprehend the information presented by the graph and make any necessary modifications to enhance its interpretability. The goal is to extract meaningful insights from the graph that can inform decision-making or provide valuable information about the stock exchange data.
In this context, anomaly detection plays a significant role. One of the anomaly detection algorithms described earlier [move to that page] will be selected for this purpose. By applying the chosen algorithm to the displayed graph, changes will be made to highlight and visualize any detected anomalies. This allows for a deeper exploration of the data and facilitates the identification and understanding of unusual or abnormal patterns that may exist within the stock exchange data.
By incorporating anomaly detection algorithms into the analysis and visualization process, the graphical representation can be adjusted to better highlight and emphasize anomalous data points, aiding in the identification of potential risks, outliers, or irregularities in the stock exchange data. This integration of anomaly detection techniques enhances the overall analysis and provides valuable insights for decision-making in the realm of stock trading and market analysis.

2.8.1. Anomaly Detection in Project

Two different algorithms were used for anomaly detection. one of them is the IQR algorithm, and the other is the iForest algorithm included in the PyOD package. Now

let's look at why these algorithms are used and how they are implemented.

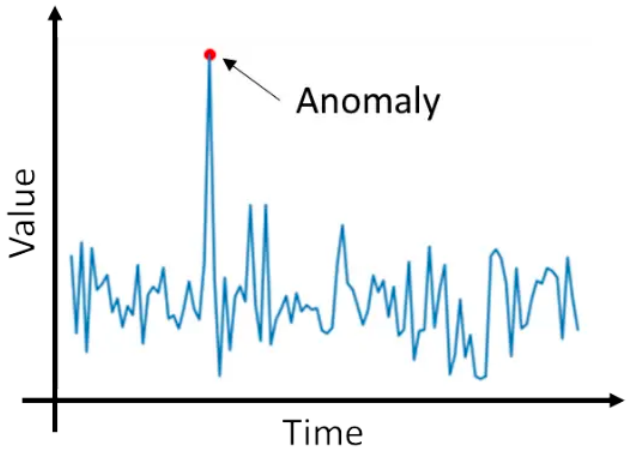

Figure 2.4: An Anomaly point in a Graph [26]

2.8.1.1. Interquartile Range (IQR)

The simplest approach to identifying irregularities in data is to flag the data points that deviate from common statistical properties of distribution, including mean, median, mode, and quartiles.
One of the most popular ways is the Interquartile Range (IQR). IQR is a concept in statistics that is used to measure the statistical dispersion and data variability by dividing the dataset into quartiles.
In simple words, any dataset or any set of observations is divided into four defined intervals based upon the values of the data and how they compare to the entire dataset. A quartile is what divides the data into three points and four intervals. [27]

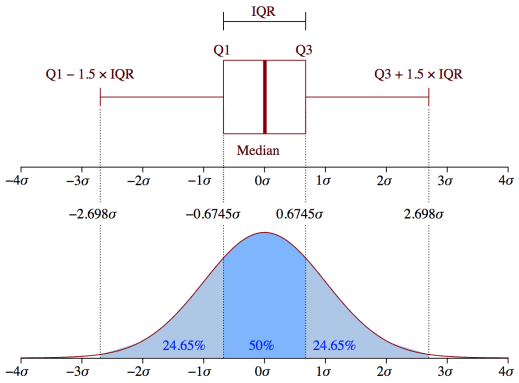

Figure 2.5: Example picture shows how IQR algorithm works [27]

2.8.2. PyOD - Python Outlier Detection

PyOD, short for Python Outlier Detection, was used in the project for its powerful outlier detection algorithms and functionalities. PyOD offers a wide range of outlier detection methods, including statistical and machine learning-based approaches. It was chosen for the project because of its ease of use, extensive documentation, and community support.

PyOD provides an intuitive and user-friendly interface, making it easy to integrate into the existing project workflow. It is optimized for scalability and performance, allowing it to handle large datasets efficiently. Additionally, PyOD offers customization options, enabling users to fine-tune the algorithms for their specific needs.

By using PyOD, the project was able to effectively detect anomalies in the stock exchange data. This enhanced the analysis process and provided valuable insights for decision-making in stock trading and market analysis.

The Isolation Forest (iForest) algorithm, available within the PyOD library, was utilized in the project for anomaly detection. The iForest algorithm is a popular and efficient method for identifying outliers in a dataset. It leverages the concept of random forests to isolate anomalies by constructing a tree-based partitioning of the data.

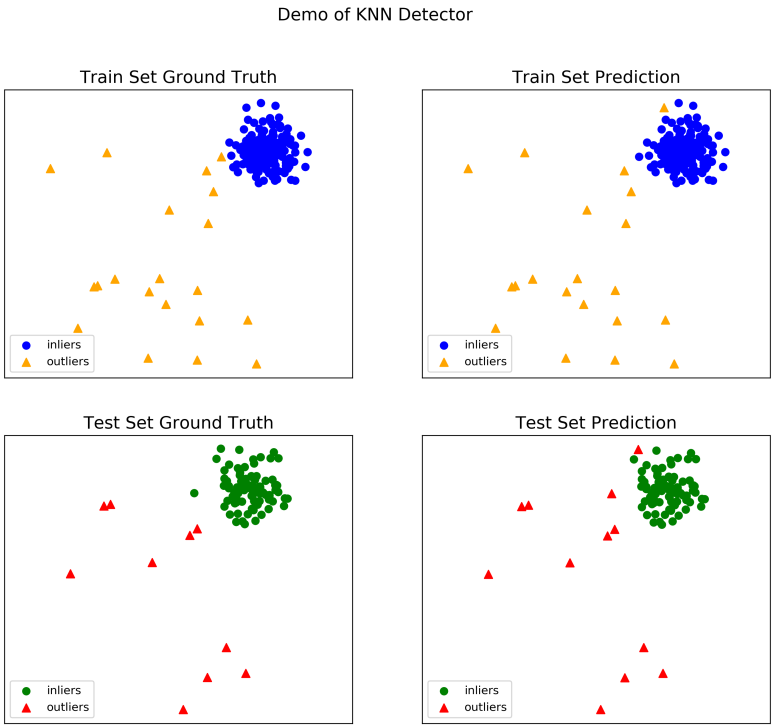

Figure 2.6: Demo of KNN Anomaly Detector [28]

2.8.2.1. Isolation Forest (iForest)[29]

Isolation Forest is an unsupervised learning algorithm that belongs to the ensemble decision trees family. This approach is different from all previous methods. All the previous ones were trying to find the normal region of the data then identifies anything outside of this defined region to be an outlier or anomalous. This method works differently. It explicitly isolates anomalies instead of profiling and constructing normal points and regions by assigning a score to each data point. It takes advantage of the fact that anomalies are the minority data points and that they have attribute-values that are very different from those of normal instances.

IsolationForest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.
Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.
This path length averaged over a forest of such random trees, is a measure of normality and our decision function.

Random partitioning produces noticeable shorter paths for anomalies. Hence, when a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies.
This algorithm works great with very high dimensional datasets and it proved to be a very effective way of detecting anomalies.

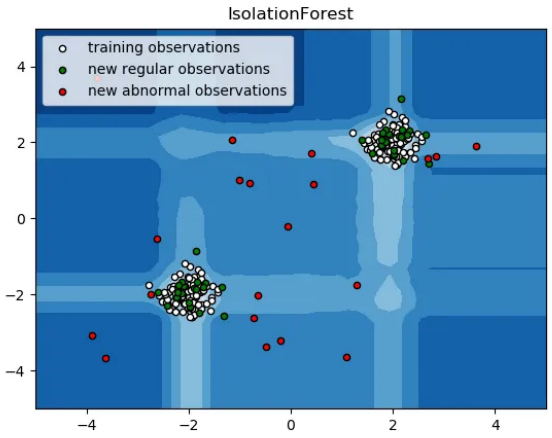

Figure 2.7: iForest algorithm visualization example for randomly trained data [30]

2.8.2.2. Why iForest is the best anomaly detection algorithm for big data?

First of all, it is necessary to explain the meanings of some of the abbreviations that will be used shortly. ROC or Receiver Operating Characteristic plot is used to visualise the performance of a binary classifier. It gives us the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at different classification thresholds. Preciison is the quality of a positive prediction made by the model. Best-in-class performance that generalizes. iForest performs better than most other outlier detection (OD) algorithms across a variety of datasets, based on ROC performance and Precision. Benchmark data has taken from the authors of the Python Outlier Detection package (PyOD; original paper) and applied row-wise green-red gradient conditional formatting in Excel. Dark green indicates the best performing algorithm for a dataset and dark red indicates the worst performing:

*conditional formatting is a row-wise comparison

ROC Performances (average of 10 independent trials)

Data	#Samples	# Dimensions	Outlier Perc	ABOD	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA
arrhythmia	452	274	14.60	0.7688	0.7835	0.7781	0.8219	0.8005	0.7861	0.7787	0.779	0.7812	0.7815
cardio	1,831	21	9.61	0.5692	0.9276	0.5867	0.8351	0.9213	0.7236	0.5736	0.8135	0.9348	0.9504
glass	214	9	4.21	0.7951	0.8504	0.8726	0.7389	0.7569	0.8508	0.8644	0.7901	0.6324	0.6747
ionosphere	351	33	35.90	0.9248	0.8134	0.873	0.5614	0.8499	0.9267	0.8753	0.9557	0.8419	0.7962
letter	1,600	32	6.25	0.8783	0.507	0.866	0.5927	0.642	0.8766	0.8594	0.8074	0.6118	0.5283
lympho	148	18	4.05	0.911	0.9728	0.9753	0.9957	0.9941	0.9745	0.9771	0.9	0.9759	0.9847
mnist	7,603	100	9.21	0.7815	0.8009	0.7205	0.5742	0.8159	0.8481	0.7161	0.8666	0.8529	0.8527
musk	3,062	166	3.17	0.1844	0.9879	0.5263	1	0.9999	0.7986	0.5287	0.9998	1	1
optdigits	5,216	64	2.88	0.4667	0.5089	0.4434	0.8732	0.7253	0.3708	0.45	0.3979	0.4997	0.5086
pendigits	6,870	16	2.27	0.6878	0.9486	0.4595	0.9238	0.9435	0.7486	0.4698	0.8344	0.9303	0.9352
pima	768	8	34.90	0.6794	0.7348	0.6235	0.7	0.6806	0.7078	0.6271	0.6753	0.6215	0.6481
satellite	6,435	36	31.64	0.5714	0.6693	0.5572	0.7581	0.7022	0.6836	0.5573	0.803	0.6622	0.5988
satimage-2	5,803	36	1.22	0.819	0.9917	0.457	0.9804	0.9947	0.9536	0.4577	0.9959	0.9978	0.9822
shuttle	49,097	9	7.15	0.6234	0.6272	0.4724	0.9855	0.9971	0.6537	0.5264	0.9903	0.9917	0.9898
vertebral	240	6	12.50	0.4262	0.3486	0.4166	0.3263	0.3905	0.3817	0.4081	0.3906	0.4431	0.4027
vowels	1,456	12	3.43	0.9606	0.5856	0.9425	0.6727	0.7585	0.968	0.941	0.8076	0.7802	0.6027
wbc	378	30	5.56	0.9047	0.9227	0.9325	0.9516	0.931	0.9366	0.9349	0.921	0.9319	0.9159
			mean	0.7031	0.7636	0.6767	0.7819	0.8179	0.7758	0.6792	0.8075	0.7935	0.7737
			median	0.7688	0.8009	0.6235	0.8219	0.8159	0.7986	0.6271	0.8135	0.8419	0.7962
			sd	0.2038	0.1890	0.1966	0.1866	0.1590	0.1764	0.1929	0.1738	0.1775	0.1916

Figure 2.8: Benchmark data has taken from PyOD (Python Outlier Detection toolbox) and applied row-scope green-red gradient conditional formatting in Excel. Green means “good” and red means “bad.” We see that iForest leads across most datasets and in aggregate, as shown by mean, median and standard deviation colors. [31]

We see that iForest leads across most datasets and in aggregate, as shown by colors for the mean, median and standard deviation rows that I calculated. The same pattern of superior results for iForest holds true for precision @ N performances:

*conditional formatting is a row-wise comparison

Precision @ N Performances (average of 10 independent trials)

Data	#Samples	# Dimensions	Outlier Perc	ABOD	CBLOF	FB	HBOS	IForest	KNN	LOF	MCD	OCSVM	PCA
arrhythmia	452	274	14.60	0.3808	0.4539	0.4230	0.5111	0.4961	0.4464	0.4334	0.3995	0.4614	0.4613
cardio	1,831	21	9.61	0.2374	0.5876	0.1690	0.4476	0.5041	0.3323	0.1541	0.4317	0.5011	0.6090
glass	214	9	4.21	0.1702	0.0726	0.1476	0.0000	0.0726	0.0726	0.1476	0.0000	0.1726	0.0726
ionosphere	351	33	35.90	0.8442	0.6088	0.7056	0.3295	0.6369	0.8602	0.7063	0.8806	0.7000	0.5729
letter	1,600	32	6.25	0.3801	0.0749	0.3642	0.0715	0.1003	0.3312	0.3641	0.1933	0.1510	0.0875
lympho	148	18	4.05	0.4483	0.7517	0.7517	0.8467	0.9267	0.7517	0.7517	0.5183	0.7517	0.7517
mnist	7,603	100	9.21	0.3555	0.3348	0.3299	0.1188	0.3135	0.4204	0.3343	0.3462	0.3962	0.3846
musk	3,062	166	3.17	0.0507	0.7766	0.2230	0.9783	0.9680	0.2733	0.1695	0.9742	1.0000	0.9799
optdigits	5,216	64	2.88	0.0060	0.0000	0.0244	0.2194	0.0301	0.0000	0.0234	0.0000	0.0000	0.0000
pendigits	6,870	16	2.27	0.0812	0.2768	0.0658	0.2979	0.3422	0.0984	0.0653	0.0893	0.3287	0.3187
pima	768	8	34.90	0.5193	0.5413	0.4480	0.5424	0.5111	0.5413	0.4555	0.4962	0.4704	0.4943
satellite	6,435	36	31.64	0.3902	0.4152	0.3902	0.5690	0.5676	0.4994	0.3893	0.6845	0.5346	0.4784
satimage-2	5,803	36	1.22	0.2130	0.8846	0.0555	0.6939	0.8754	0.3809	0.0555	0.6481	0.9356	0.8041
shuttle	49,097	9	7.15	0.1977	0.2943	0.0695	0.9551	0.9546	0.2184	0.1424	0.7506	0.9542	0.9501
vertebral	240	6	12.50	0.0601	0.0000	0.0644	0.0071	0.0343	0.0238	0.0506	0.0071	0.0238	0.0226
vowels	1,456	12	3.43	0.5710	0.0831	0.3224	0.1297	0.1875	0.5093	0.3551	0.2186	0.2791	0.1364
wbc	378	30	5.56	0.3060	0.5055	0.5188	0.5817	0.5088	0.4952	0.5188	0.4577	0.5125	0.4767
			mean	0.3066	0.3919	0.2984	0.4294	0.4723	0.3679	0.3010	0.4174	0.4808	0.4471
			median	0.3060	0.4152	0.3224	0.4476	0.5041	0.3809	0.3343	0.4317	0.4704	0.4767
			sd	0.2109	0.2740	0.2186	0.3099	0.3159	0.2339	0.2190	0.2968	0.3011	0.3039

Figure 2.9: [32]

3. PYTHON IMPLEMENTATION AND APPLICATION

3.1. Requirements

Before running the source code, there are several Python libraries and packages that need to be installed as dependencies. The following Python bootstraps are required:

1. Python 3.6: The programming language used for the project.
2. numpy: A library for numerical computing in Python, used for efficient array operations.
3. pandas: A library for data manipulation and analysis, used for handling and processing datasets.
4. matplotlib 2.2: A plotting library for creating visualizations, used for generating graphs and charts.
5. scipy: A scientific computing library, used for various mathematical and statistical computations.
6. PyQt 5.9.2: A Python binding for the Qt framework, used for creating the GUI.
7. Python 3 TK: The TKinter library for Python 3, used for creating the user interface components.

These bootstraps need to be installed and configured properly in the Python environment before running the source code to ensure that all required dependencies are available for the program to execute successfully.

3.2. Installation

1. Install Python 3.6

```
$ sudo apt-get install python3.6
```

2. Install pip3

```
$ sudo apt-get install python3-pip
```

3. Install Numpy

```
$ pip install numpy
```

4. Install Pandas

```
$ pip install pandas
```

5. Install Matplotlib

```
$ pip install matplotlib
```

6. Install PyQT5

```
$ pip install PyQt5==5.9.2
```

7. Install Python 3 TK

```
$ sudo apt-get install python3-tk
```

8. Install Calendar

```
$ pip install calendar
```

9. Install tkcalendar

```
$ pip install tkcalendar
```

10. Install pywt

```
$ pip install pywt
```

11. Install pyOD

```
$ pip install pyod
```

12. Install mplcursors

```
$ pip install mplcursors
```

3.3. Project Files

There are two type of python file in this project. First one (grad_prod) used to get Borsa Istanbul's Intraday Transaction Book Information and collect all the data in a ".csv" files with their date. Second one (grad_cons) used to process the data with Fourier Transform and plot the results.

Note: constants.py includes all the constants that used in both grad_prod and grad_cons, it includes holiday days and their calculations.

3.3.1. grad_prod.py

This python file can be used to convert PP_GUNICIISLEM.M.[year][month].csv files to collection of .csv files with their date. It uses Borsa Istanbul's Intraday Transaction Book Information. It can be used with following command:

```
$ python3 grad_prod.py
```

After transform process, it will create a folder named "data[year][month]" and put all the .csv files in it. .csv files will be named as [year]-[month]-[day].csv.

These .csv files will include total volume that is traded in that second. It will also include the date and time of that second. It will also include the total volume that is traded in that second. By using these .csv files, we can process the data with Fourier Transform, Visualization of Data and process it to be used for multiple actions and Anomaly Detection.

3.3.2. grad_cons.py

This python file can be used to process the data with Fourier Transform and plot the results. It can be used with following command:

```
$ python3 grad_cons.py
```

After compilation, it will open a tkinter window. In this window, there will be a Calendar of 2023 that can be selected the date, two radio button components, one is for selecting the type of the graph which are Time, Frequency and Wavelet Graphs. Other one is for selecting the type of the algorithm that will be used for detect anomalies. There are 2 algorithms implemented to detect anomalies which are IForest and the other one is IQR. There are two boxes that will be used to limit the time interval of the graph. There is a slider for get the outlier detection contamination from user and "Plot Graph" button to plot the graph.

After selecting the date, type of the graph, type of the algorithm, time interval and outlier detection contamination, user can click "Plot Graph" button to plot the graph. After plotting the graph, we will be able to see the visualization of the data and the anomalies that are detected by the algorithm.

Other than that, In time and frequency graphs, the time that is detected by the anomaly detection algorithm is printed to the command line, so that we can see the time of the anomaly that is occured.

3.4. Flow of project

3.4.1. User Interface

The program presents a user interface created with Tkinter upon opening. The interface offers various options to customize the chart that will be generated. Here is a breakdown of the components in the interface:

1. Calendar: A calendar is displayed at the top, allowing the user to select any day from the year 2023. This selection determines the date for which the chart will be generated.

2. Transformation Type: Below the calendar, there are three radio buttons representing different transformation types. The user can choose one of these options to specify the type of transformation to be applied to the data for charting.

3. Anomaly Detection Algorithm: In the algorithm section, there are two options for anomaly detection algorithms. On the left side, the iForest algorithm from PyOD (as explained in previous sections) is available, while on the right side, the IQR algorithm (also explained earlier) is provided. The user can choose one of these algorithms for anomaly detection.

4. Time Range Selection: At the bottom, there are two input boxes for setting the time range. The left box is for selecting the hour, and the right box is for selecting the minutes. These inputs determine the time range for both charting and anomaly detection.

5. Outlier Coefficient: A slider is placed to determine the outlier coefficient. This slider ranges from 0.1 to 10, and the user can adjust it to set the desired coefficient value.

6. Plot Graph Button: At the bottom, there is a button labeled ”Plot Graph.” When this button is clicked, the program generates the chart based on the selected options and displays it.

Overall, the user interface provides a convenient way to customize the chart and perform anomaly detection by selecting the desired date, transformation type, algorithm, time range, and outlier coefficient. The ”Plot Graph” button triggers the generation of the chart with the selected settings.

3.4.2. Graph Generation

After the graph is generated, it will display red dots and blue lines overlaid on the chart. The specific graph type will depend on the transformation type selected by the user.
If the user chooses the Time or Frequency domain transformation, the graph will be presented as a scatter plot, with red dots indicating anomalies and blue lines representing the original data points. The anomalies detected by the selected anomaly detection algorithm will be marked with red dots, helping to visually distinguish them from the rest of the data points.
On the other hand, if the user selects the Wavelet Transform, the graph will be presented as a line plot. The blue lines will represent the original data points, and the red dots will indicate the detected anomalies.
In the next section, screenshots of these different graph types will be included to provide a visual representation of the anomaly detection results.

3.5. Screenshots

3.5.1. User Interface

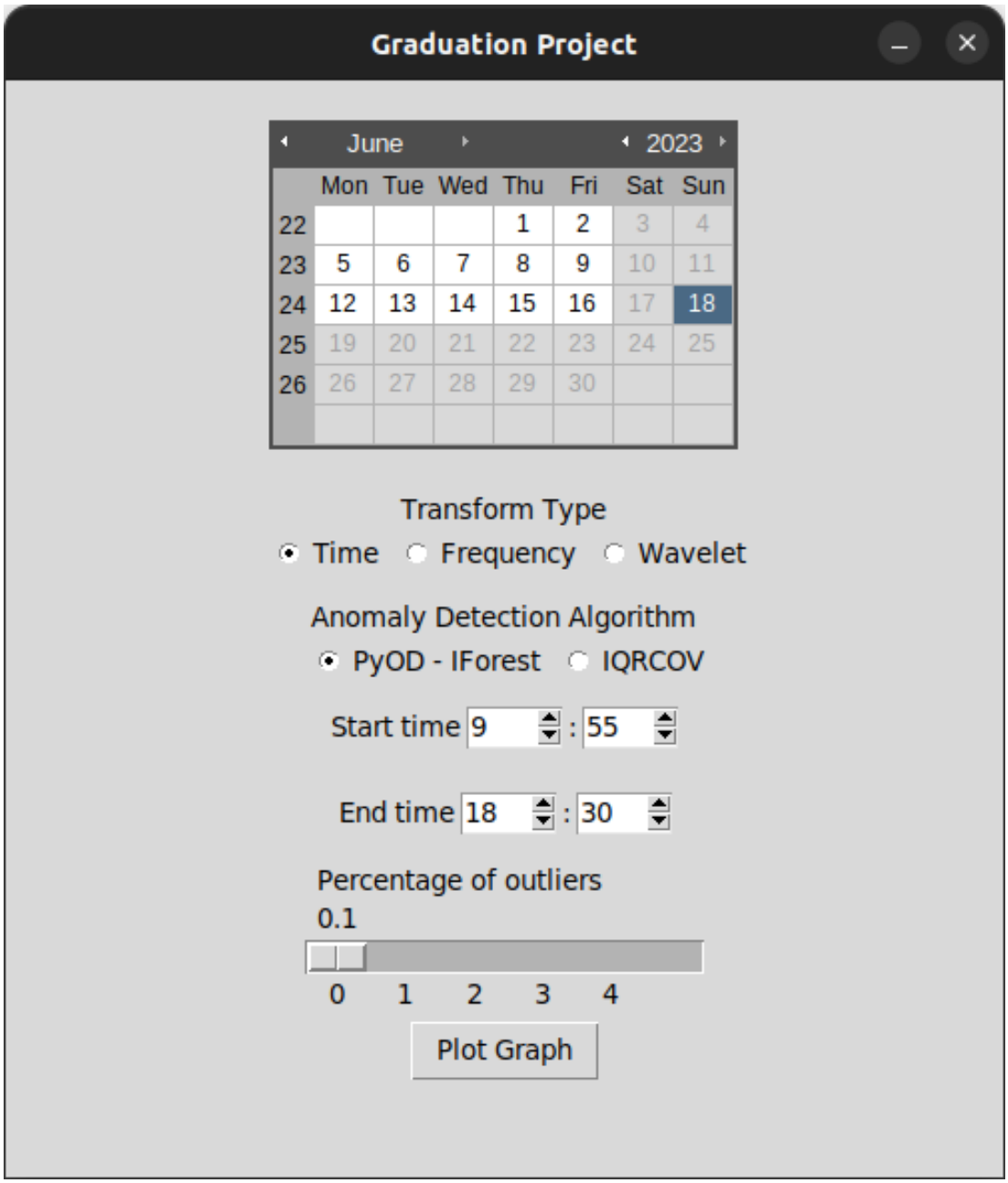

Figure 3.1: User Interface screen where user's encounter first

3.5.2. Time Graphs

3.5.2.1. iForest Algorithm

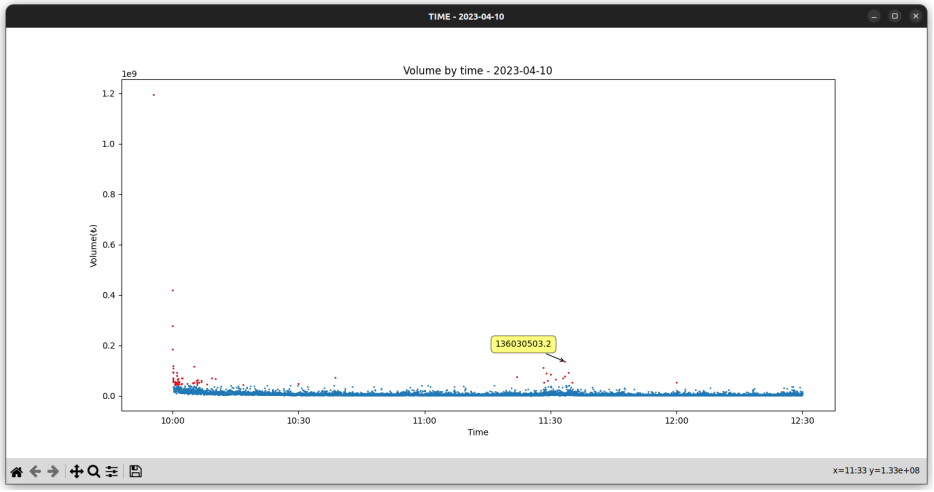

Figure 3.2: Time Graph at 10 April 2023 from start time 9:55 - end time 12:30 and 0.7 percentage with iForest Anomaly Detection Algorithm

3.5.2.2. IQR Algorithm

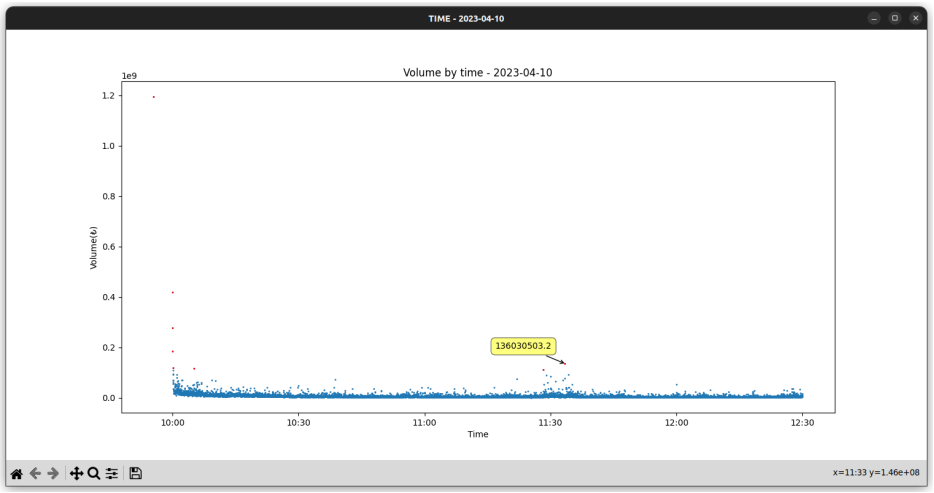

Figure 3.3: Time Graph at 10 April 2023 from start time 9:55 - end time 12:30 and 0.7 percentage with IQR Anomaly Detection Algorithm

3.5.3. Frequency Graphs

3.5.3.1. iForest Algorithm

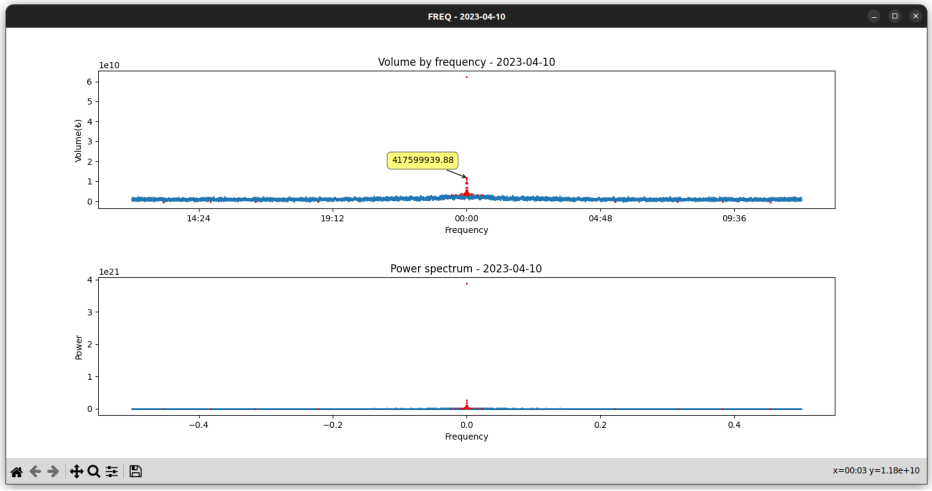

Figure 3.4: Frequency Graph at 10 April 2023 from start time 9:55 - end time 12:30 and 0.7 percentage with iForest Anomaly Detection Algorithm

3.5.3.2. IQR Algorithm

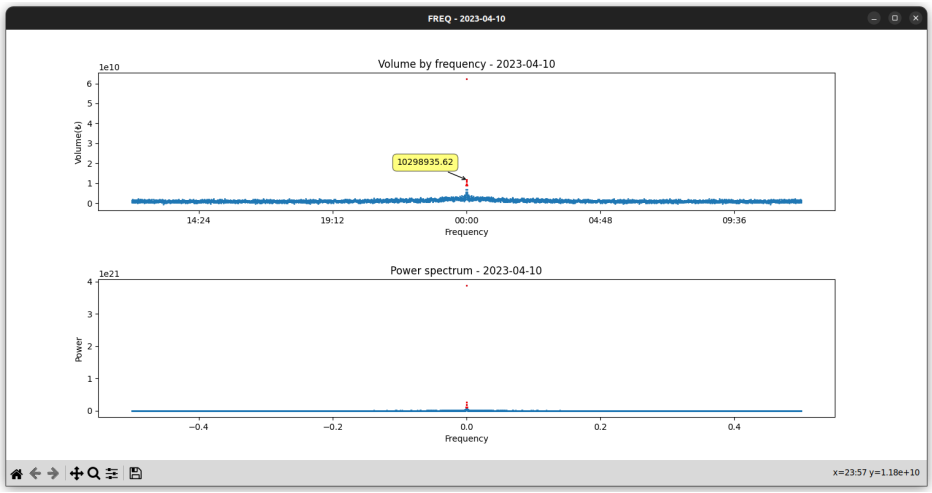

Figure 3.5: Frequency Graph at 10 April 2023 from start time 9:55 - end time 12:30 and 0.7 percentage with IQR Anomaly Detection Algorithm

3.5.4. Wavelet Graphs

3.5.4.1. iForest Algorithm

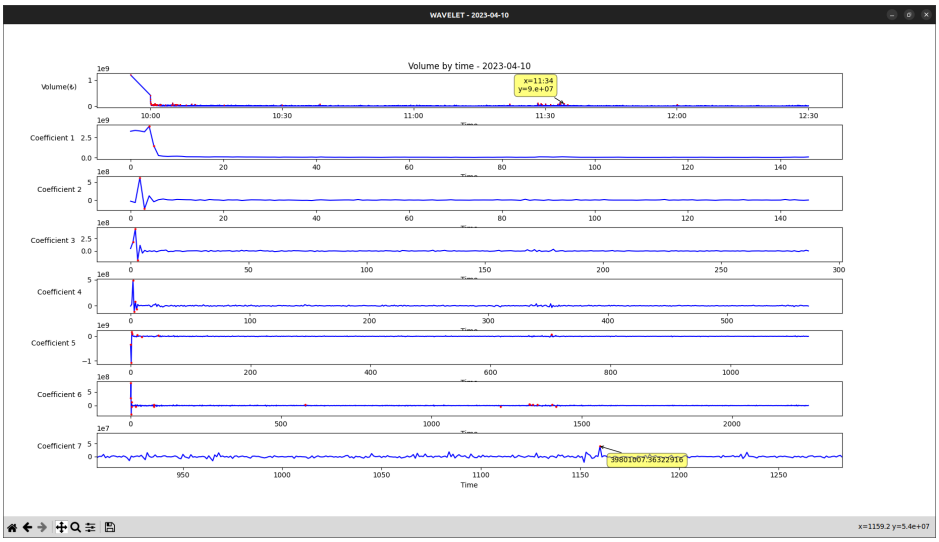

Figure 3.6: Wavelet Graphs at 10 April 2023 from start time 9:55 - end time 12:30 and 0.7 percentage with iForest Anomaly Detection Algorithm

3.5.4.2. IQR Algorithm

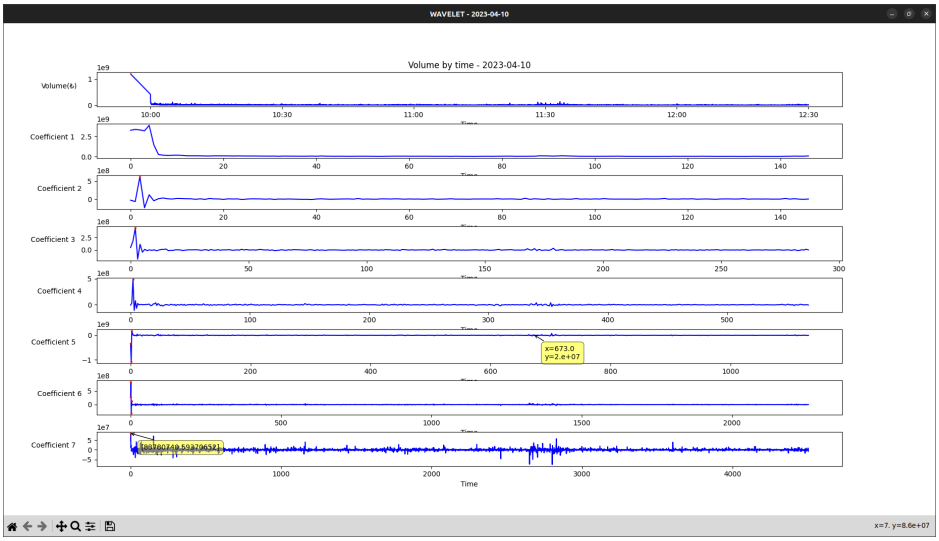

Figure 3.7: Wavelet Graphs at 10 April 2023 from start time 9:55 - end time 12:30 and 0.7 percentage with IQR Anomaly Detection Algorithm

3.6. Anomaly Detection Graph Analysis

The analysis of anomalies in a created time graph typically involves the following steps:

1. Visual Inspection: Begin by visually inspecting the time graph to identify any noticeable deviations, patterns, or irregularities that may indicate the presence of anomalies.

2. Define Normal Behavior: Establish a baseline or expectation of what constitutes normal behavior in the time graph. This can be done by analyzing historical data or establishing statistical measures such as mean, median, or standard deviation to define the normal range of values.

3. Identify Anomalies: Compare the observed data points in the time graph against the defined normal behavior. Identify data points that fall outside the expected range or exhibit abnormal patterns.

4. Contextual Analysis: Conduct a contextual analysis to understand the underlying factors contributing to the detected anomalies. Consider temporal factors, external influences, data patterns, domain expertise, historical context, and any auxiliary data that may provide insights into the anomalies.

5. Validation: Validate the detected anomalies to ensure their accuracy and relevance. This can involve cross-referencing with independent sources of information, consulting subject matter experts, or conducting further analysis using complementary techniques.

6. Impact Assessment: Assess the impact of the detected anomalies on the overall data or the system under consideration. Evaluate the significance of the anomalies in terms of their potential consequences, risks, or implications for decision-making.

7. Action and Decision-making: Based on the analysis and assessment of the anomalies, determine the appropriate actions or decisions to be taken. This could involve anomaly resolution, data cleansing, system adjustments, or further investigations.

3.7. Trailer Video

If you want to see the demo of the project, you can click here to go to the youtube video that trailer of this project.

4. CONCLUSIONS

In conclusion, the developed system offers a user-friendly interface that allows users to select multiple graph types, such as time domain, frequency domain, and wavelet domain, for visualizing the data. This flexibility empowers users to gain different perspectives and insights into the underlying patterns and anomalies present in the dataset. Furthermore, the system provides the option to choose between multiple anomaly detection algorithms, namely the Isolation Forest (iForest) algorithm and the Interquartile Range (IQR) algorithm. This feature enables users to compare and evaluate the performance of different algorithms in detecting anomalies within the selected graph types. By incorporating these features, the system empowers users to conduct comprehensive analyses, leveraging various graph types and algorithms to uncover anomalies effectively. This flexibility and adaptability enhance the accuracy and reliability of anomaly detection, enabling users to make informed decisions and take appropriate actions based on the identified anomalies. Overall, the system offers a powerful toolkit for anomaly detection, enabling users to explore data from different perspectives, apply multiple algorithms, and gain valuable insights for decision-making and anomaly management.

BIBLIOGRAPHY

[1] Wikipedia, *Fourier transform — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Fourier%20transform&oldid=1160462432, [Online; accessed 18-June-2023], 2023.

[2] Wikipedia, *Laplace transform — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Laplace%20transform&oldid=1158213292, [Online; accessed 18-June-2023], 2023.

[3] Wikipedia, *Z-transform — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Z-transform&oldid=1160407569, [Online; accessed 18-June-2023], 2023.

[4] Wikipedia, *Wavelet transform — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Wavelet%20transform&oldid=1147185762, [Online; accessed 18-June-2023], 2023.

[5] Wikipedia, *Frequency domain — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Frequency%20domain&oldid=1146959234, [Online; accessed 18-June-2023], 2023.

[6] E. Chu, *Discrete and Continuous Fourier Transforms: Analysis, Applications and Fast Algorithms*. Mar. 2008, ISBN: 9780429144752. DOI: 10.1201/9781420063646.

[7] May 2023. [Online]. Available: https://en.wikipedia.org/wiki/Discrete_Fourier_transform.

[8] Wikipedia, *Fast Fourier transform — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Fast%20Fourier%20transform&oldid=1158137318, [Online; accessed 18-June-2023], 2023.

[9] M. Gupta, J. Gao, C. Aggarwal, and J. Han, *Outlier detection for Temporal Data*, 1st ed. Springer Cham, vol. 1.

[10] Wikipedia, *Standard score — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Standard%20score&oldid=1147224272, [Online; accessed 18-June-2023], 2023.

[11] Wikipedia, *Percentile — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Percentile&oldid=1158049512, [Online; accessed 18-June-2023], 2023.

[12] Wikipedia, *Interquartile range — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Interquartile%20range&oldid=1155015588, [Online; accessed 18-June-2023], 2023.

[13] D. Williamson, R. Parker, and J. Kendrick, "The box plot: A simple visual method to interpret data," *Annals of internal medicine*, vol. 110, pp. 916–21, Jul. 1989. DOI: 10.1059/0003-4819-110-11-916.

[14] Q. Liu and Y. Wu, "Supervised learning," Jan. 2012. DOI: 10.1007/978-1-4419-1428-6_451.

[15] *What is Unsupervised Learning? — IBM — ibm.com*, https://www.ibm.com/topics/unsupervised-learning, [Accessed 18-Jun-2023].

[16] Y. Reddy, V. Pulabaigari, and E. B, "Semi-supervised learning: A brief review," *International Journal of Engineering Technology*, vol. 7, p. 81, Feb. 2018. DOI: 10.14419/ijet.v7i1.8.9977.

[17] *Anomaly Detection in Machine Learning — serokell.io*, https://serokell.io/blog/anomaly-detection-in-machine-learning, [Accessed 18-Jun-2023].

[18] Wikipedia, *Autoregressive integrated moving average — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Autoregressive%20integrated%20moving%20average&oldid=1144826256, [Online; accessed 18-June-2023], 2023.

[19] Wikipedia, *Exponential smoothing — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Exponential%20smoothing&oldid=1151784266, [Online; accessed 18-June-2023], 2023.

[20] *How Change Point Detection worksx2014;ArcGIS Pro — Documentation — pro.arcgis.com*, https://pro.arcgis.com/en/pro-app/latest/tool-reference/space-time-pattern-mining/how-change-point-detection-works.htm, [Accessed 18-Jun-2023].

[21] M. Alam, *K-Nearest Neighbors (kNN) for anomaly detection — towardsdatascience.com*, https://towardsdatascience.com/k-nearest-neighbors-knn-for-anomaly-detection-fdf8ee160d13, [Accessed 18-Jun-2023].

[22] Wikipedia, *Local outlier factor — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Local%20outlier%20factor&oldid=1152220901, [Online; accessed 18-June-2023], 2023.

[23] *Exploratory Data Analysis in R — kaggle.com*, https://www.kaggle.com/code/heeraldedhia/exploratory-data-analysis-in-r, [Accessed 18-Jun-2023].

[24] S. Gürsakal, "Frequency domain approach to financial time series analysis: Wavelets methods and an evidence from İmkb," Available at https://acikerisim.uludag.edu.tr/bitstream/11452/3353/1/240867.pdf, PhD thesis, Uludağ Üniversitesi, Example City, CA, 2009.

[25] *PyWavelets CWT implementation — dsp.stackexchange.com*, https://dsp.stackexchange.com/questions/70575/pywavelets-cwt-implementation, [Accessed 18-Jun-2023].

[26] S. R, *A walkthrough of Univariate Anomaly Detection in Python — analyticsvidhya.com*, https://www.analyticsvidhya.com/blog/2021/06/univariate-anomaly-detection-a-walkthrough-in-python/, [Accessed 18-Jun-2023].

[27] K. Srinath, *Anamoly Detection: Techniques to detect outliers — towardsdatascience.com*, https://towardsdatascience.com/anamoly-detection-techniques-to-detect-outliers-fea92047a222, [Accessed 18-Jun-2023].

[28] *GitHub - yzhao062/pyod: A Comprehensive and Scalable Python Library for Outlier Detection (Anomaly Detection) — github.com*, https://github.com/yzhao062/pyod, [Accessed 18-Jun-2023].

[29] F. T. Liu, K. Ting, and Z.-H. Zhou, "Isolation forest," Jan. 2009, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[30] K. Srinath, *Anamoly detection: Techniques to detect outliers*, Jun. 2020. [Online]. Available: https://towardsdatascience.com/anamoly-detection-techniques-to-detect-outliers-fea92047a222.

[31] *Benchmarks - pyod 1.0.9 documentation — pyod.readthedocs.io*, https://pyod.readthedocs.io/en/latest/benchmark.html, [Accessed 18-Jun-2023].

[32] A. Young, *Isolation Forest is the best Anomaly Detection Algorithm for Big Data Right Now — towardsdatascience.com*, https://towardsdatascience.com/isolation-forest-is-the-best-anomaly-detection-algorithm-for-big-data-right-now-e1a18ec0f94f, [Accessed 18-Jun-2023].