Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

Department of Electronics and Communication Engineering

III IA EVALUATION REPORT

for

BLENDED LEARNING PROJECT BASED LEARNING

A report submitted by

Name of the Students	***KEVIN PHILIP ABRAHAM & ANGEL MARIYA***
Register Numbers	***UTK22EC1001 & URK22EC2011***
Subject Name	***DIGITAL COMMUNICATION***
Subject Code	***22EC2016***
Date of Report submission	

Project Rubrics for Evaluation

First Review: to be awarded for **10 Marks** – PPT should have four slides (Title page, Introduction, Circuit/Block Diagram, and Description of Project).

Second Review: to be awarded for **10 Marks** – PPT should have three slides (Description of Circuit/Concept, Design of Circuit in Tools/Design of Flow graph, and Partial Results)

Third Review: to be awarded for **10 Marks** – PPT should have two slides (Tool based simulation/Hardware based simulation/Concept based interaction/ Case Study)

Fourth Review: to be awarded for **10 Marks** –PPT should have two slides (Output Results & Discussion)

[NOTE: The instructions are subjected to change if required to combine either of two reviews and to be evaluated for 20 Marks.]

Total marks: _____/ 40 Marks

Signature of Faculty with date:

IMPLEMENTATION OF SHANNON FANO CODING IN AN ECG SIGNAL

a project report submitted by

KEVIN PHILIP ABRAHAM(UTK22EC1001)

ANGEL MARIYA(URK22EC2011)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

For

DIGITAL COMMUNICATION

– PROJECT BASED LEARNING (22EC2016)

under the supervision of

Dr. S. Merlin Gilbert Raj

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Deemed-to-be-University)

Karunya Nagar, Coimbatore - 641 114. INDIA

October 2024

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **“IMPLEMENTATION OF SHANNON FANO CODING IN AN ECG SIGNAL”** is the bonafide work of **“KEVIN PHILIP ABRAHAM (UTK22EC1001) and ANGEL MARIYA-(URK22EC2011)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
DR.JUDE HEMANTH	DR.S.MERLIN GILBERT RAJ
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of ECE	Department of ECE
School of E&T	School of E&T

Submitted for the III Internal-Project Based Learning Viva Voce held on _______________.

Internal Examiner

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
1.	INTRODUCTION	5
2.	PROBLEM STATEMENT & GRAPHICAL ABSTRACT	6
3.	METHODOLOGY FOR SHAANNON FANO IMPLEMENTATION	9
4.	KEY PRINCIPLES INVOLVED	12
5.	DESIGN & MATHEMATICAL MODEL	16
6.	BLOCK DIAGRAM / FLOW CHART	22
7.	PYTHON CODE	23
8.	RESULT WITH GRAPH	30
9.	CONCLUSION	33
10.	REFERENCE	34

CHAPTER 1

INTRODUCTION

Electrocardiography (ECG) is a widely used diagnostic tool that measures and records the electrical activity of the heart. The ECG signal provides valuable insights into cardiac health, helping in the detection of various heart conditions such as arrhythmias, ischemia, and other abnormalities. However, ECG signals can generate a large amount of data, especially in long-term monitoring or high-resolution applications. Efficient compression of ECG signals is essential to reduce data storage, transmission bandwidth, and computational overhead without compromising the diagnostic quality of the signal.One popular approach for data compression is the use of entropy-based encoding methods, such as the Shannon-Fano algorithm. The Shannon-Fano method is a lossless compression technique that efficiently encodes data based on the probability distribution of the source symbols. The method works by assigning shorter codes to more frequent symbols and longer codes to less frequent symbols, thereby achieving compression.

In this project, we explore the application of the Shannon-Fano encoding algorithm to ECG signals. The goal is to compress the ECG data in a manner that preserves critical information required for accurate diagnosis while significantly reducing the file size. This is particularly important in telemedicine, where ECG data needs to be transmitted over limited-bandwidth channels or stored for extended periods.The implementation of the Shannon-Fano algorithm in ECG signal encoding involves several key steps, including preprocessing the ECG signal, determining the symbol probability distribution, constructing the Shannon-Fano code tree, and evaluating the compression efficiency. By comparing the compression ratio and signal fidelity, this project aims to demonstrate the effectiveness of Shannon-Fano encoding in ECG data compression and provide insights into its potential use in medical applications.

In summary, this project seeks to address the challenges associated with ECG signal compression by leveraging the Shannon-Fano encoding technique, offering a balance between compression efficiency and the preservation of clinically relevant information.

CHAPTER-2

PROBLEM STATEMENT & GRAPHICAL ABSTRACT

- **Problem Statement**

Electrocardiogram (ECG) signals are critical for diagnosing heart conditions, capturing complex and often subtle patterns like P waves, QRS complexes, and T waves that must be represented with high precision. ECG data is typically generated in large volumes, making efficient storage and transmission a challenge, especially in telemedicine and wearable health monitoring devices. At the same time, any data compression technique must preserve the precision of the signal to ensure that important medical information, such as indications of arrhythmia or ischemia, is not lost.

Shannon-Fano coding, an entropy-based lossless data compression technique, can reduce the size of ECG data by assigning variable-length binary codes to signal values based on their frequency of occurrence. However, the inherent complexity and variability of ECG signals require a compression method that can handle the precision needed for accurate diagnosis while maintaining efficiency.

Objectives:

Input Data: Process a digitized ECG signal with high precision (e.g., millivolts sampled at regular intervals) to retain diagnostically critical features.

Symbol Construction: Divide the ECG signal into discrete value ranges (symbols) and compute their respective frequencies, maintaining a high degree of precision.

Code Generation: Implement the Shannon-Fano coding algorithm to assign binary codes to each symbol based on its frequency of occurrence, ensuring that precise values are accurately represented.

Encoding: Use the generated codes to compress the ECG signal without losing critical diagnostic details.

Performance Evaluation: Analyze the trade-off between compression efficiency and the retention of signal precision, ensuring that the compressed ECG data can still be used for accurate medical diagnosis.

- **Graphical Abstract**

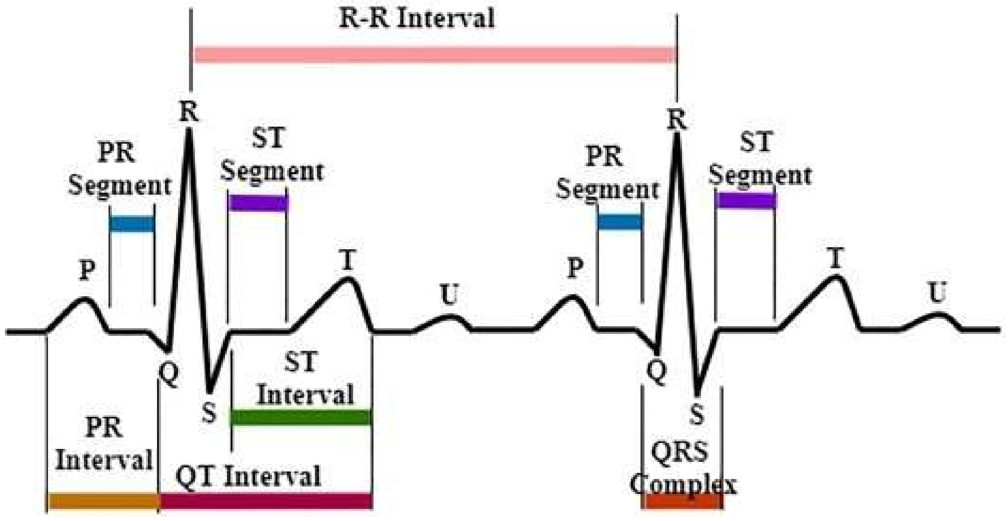


Fig: Makeup of a ECG signal

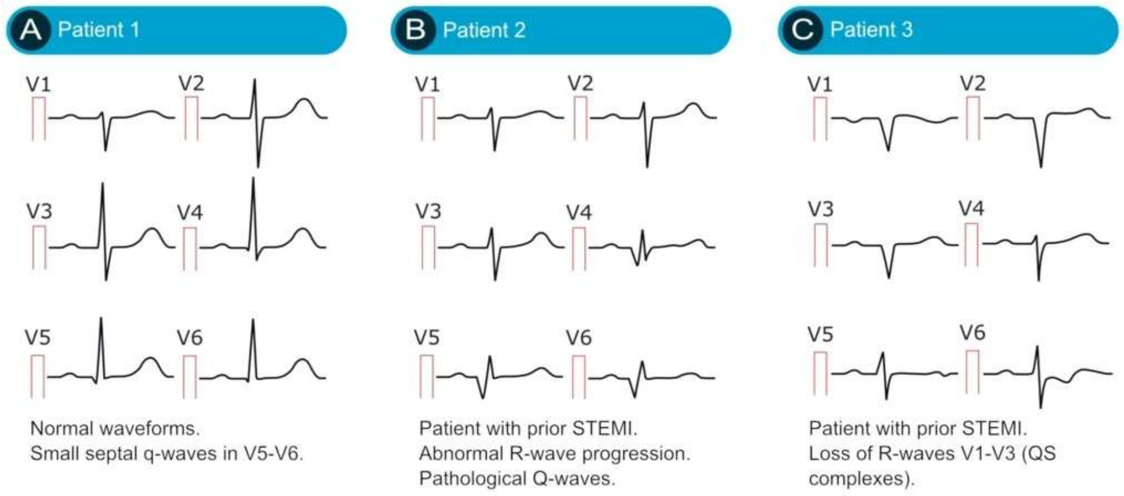

A Patient 1
B Patient 2
C Patient 3
V1
V2
V3
V4
V5
V6
Normal waveforms.
Small septal q-waves in V5-V6.
Patient with prior STEMI.
Abnormal R-wave progression.
Pathological Q-waves.
Patient with prior STEMI.
Loss of R-waves V1-V3 (QS complexes).

Fig: Normal and abnormal ECG signals

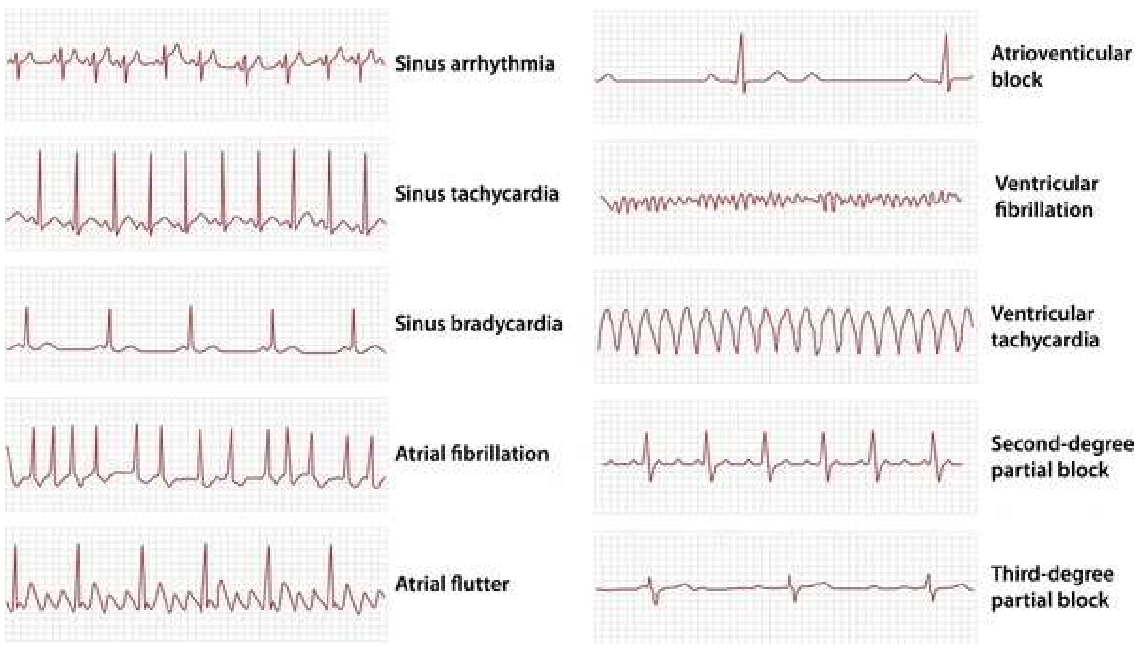


Fig: Different types of abnormalities and its characteristics in an ECG signal

Inference:

Minute variations indicate different types of abnormalities in a ECG signal hence the preservation of data is key while compressing the signal.

CHAPTER-3

METHODOLOGY FOR SHANNON FANO IMPLEMENTATION

The objective of this project is to implement a Python-based solution to compress ECG (Electrocardiogram) signals using the Shannon-Fano encoding algorithm. ECG signals are critical in monitoring heart activity, but the large volume of data they generate can be challenging to store and transmit efficiently. The Shannon-Fano method, a lossless entropy-based compression technique, provides an effective way to reduce the size of the data without compromising the integrity of the signal, which is crucial for accurate medical diagnostics.

ECG Signal Acquisition:

The project will begin by acquiring ECG signal. 20 peoples real world filtered ECG signals from PhysioNet. LW: .../1.0.0/Person_01/rec_1 (physionet.org) which is already quantized by a 12 bit ADC having a clock frequency of 500 ticks per sec, I tick per sample 200 ADU/mV, zero at 0 , base line at zero.

Implementation of Shannon-Fano Encoding:

1. **Load ECG Data:**

   Load the ADC-converted ECG signal data from PhysioNet, with a total sample size of 147,499, representing combined data from 20 people.

2. **Probability Calculation:**

   Count the occurrences of each unique ADC value in the dataset.

   Calculate the probability of each ADC value by dividing its occurrence by the total number of samples.

3. **Sort Probabilities:**

Sort the ADC values in descending order based on their probabilities.

4. **Recursive Shannon-Fano Partition:**

   Split the sorted list of ADC values into two groups such that the cumulative probabilities of each group are as equal as possible. Assign binary codes: ‘0’ to values in the first group and ‘1’ to values in the second group.

5. **Assign Binary Codes:**

   Recursively divide the groups, assigning ‘0’ and ‘1’ until every ADC value has a unique binary code.

6. **Encode the ECG Signals:**

   Replace each ADC value in the dataset with its corresponding Shannon-Fano code.

7. **Store Encoded Data:**

   Store the encoded ECG signals in an Excel file, with each row representing the encoded signals for each sample.

8. **Calculate Compression Metrics:**

   - Entropy (H)
   - Average Code Word Length
   - Efficiency (η)
   - Variance of Code Lengths (σ^2)

9. **Analyse Metrics as Number of People Increases:**

   Start with the ECG data of 1 person and incrementally add more people to the dataset.At each step, calculate the metrics: entropy, average code word length, efficiency, and variance.Record the metrics for 1 person, 2 people, and so on, up to 20 people.

10.· **Generate Graphical Results:**

Plot the following graphs:

- ➢ Entropy vs. number of people.
- ➢ Average code word length vs. number of people.
- ➢ Efficiency vs. number of people.
- ➢ Variance of code word lengths vs. number of people.

11.**Store Results in Excel:**

Save the calculated metrics and graphical results in an Excel file for further analysis. Then loaded the values in mv of filtered ecg signal to a excel sheet then loaded them into Spyder Python IDE.

Tools and Libraries:

1) Python: The primary programming language for the project.
2) NumPy: For handling numerical operations and signal processing.
3) Matplotlib: For visualizing ECG signals and compression results.
4) Counter class from the collections module: For counting the occurrences of elements in a collection, such as lists or strings.
5) Pandas: For data handling and managing ECG datasets.
6) Custom Shannon-Fano Code Implementation: For encoding the signal.

Expected Outcome:

The project will result in an effective Python implementation of the Shannon-Fano algorithm for compressing ECG signals. The method will demonstrate significant data reduction while maintaining the diagnostic quality of the ECG signal, providing a balance between compression efficiency and clinical accuracy. The project also serves as a foundation for future developments in ECG signal processing, data transmission, and storage in healthcare applications.

CHAPTER-4

KEY CONCEPTS INVOLVED

To understand the encoding of ECG signals using the Shannon-Fano method, several key concepts from signal processing, information theory, and data compression must be considered. Below are the primary concepts involved in this project:

1. Electrocardiogram (ECG) Signal

An ECG signal is a graphical representation of the electrical activity of the heart over time. It is recorded through electrodes placed on the skin, and the waveform consists of key components such as the P wave, QRS complex, and T wave.

Significance: The shape and timing of these components are used to diagnose various heart conditions. Thus, preserving the integrity of the ECG signal during compression is critical.

Characteristics: ECG signals are continuous, time-varying signals that may contain noise, baseline wander, and varying amplitudes.

2. Signal Preprocessing

Noise Filtering: ECG signals may contain high-frequency noise, power line interference, and baseline drift. Filtering techniques such as low-pass or high-pass filtering can be applied to clean the signal before compression.

Segmentation**:** The signal may be divided into segments (or symbols) that will be compressed. These symbols can be individual samples or blocks of data.

Normalization: Scaling the signal to a consistent range to simplify encoding.

3. Shannon-Fano Algorithm

Entropy Encoding**:** Shannon-Fano is an entropy-based compression algorithm that encodes data by constructing variable-length codes based on the probability distribution of symbols.

Symbols: In the context of ECG signals, a symbol can be a sample value or a set of consecutive samples.

Steps in Shannon-Fano Encoding:

Frequency Analysis: Calculate the probability of occurrence of each unique symbol in the ECG data.

Symbol Sorting: Sort symbols based on their probabilities (higher frequency symbols first).

Recursive Code Assignment: Recursively divide the sorted list into two groups of nearly equal probability and assign binary digits (0 or 1) to each division. Repeat until every symbol has a unique binary code.

Encoding: Replace each symbol in the signal with its corresponding binary code to achieve compression.

Efficiency: More frequent symbols are assigned shorter codes, while less frequent symbols get longer codes, leading to effective compression.

4. Information Theory

Entropy: In information theory, entropy represents the average amount of information produced by a source of data. It is a measure of the unpredictability or randomness of the data.

Shannon Entropy**:** Defines the minimum number of bits required to encode a source of data based on the probability distribution of its symbols. The Shannon-

Fano method leverages this concept to assign variable-length codes to symbols, minimizing the average number of bits used.

Redundancy: Redundancy in the ECG signal refers to repeated or predictable patterns in the data. The Shannon-Fano method removes this redundancy by encoding only the essential information.

5. Lossless Compression

A compression technique in which the original data can be perfectly reconstructed from the compressed data. This is crucial for medical applications like ECG signal analysis, where any loss of information may lead to misdiagnosis.

Shannon-Fano as Lossless: The Shannon-Fano algorithm is a lossless encoding method because it maps symbols to binary codes without altering the original signal content.

6. Compression Ratio

A metric used to evaluate the efficiency of a compression algorithm, typically defined as the ratio of the size of the compressed data to the size of the original data.

Importance in ECG: A high compression ratio is desired to reduce storage requirements, but it must be balanced with the need to preserve the clinical quality of the signal.

7. Python Programming

1. Libraries for ECG Processing.
2. NumPy for numerical computations.
3. Matplotlib for visualizing ECG signals.
4. Pandas for handling ECG datasets.

Custom Shannon-Fano Implementation: Python will be used to implement the Shannon-Fano algorithm from scratch, allowing full control over the encoding and decoding processes for ECG data.

8. Application in Healthcare and Telemedicine

Data Storage: Efficient ECG compression is essential for storing large amounts of patient data, especially in long-term monitoring.

Transmission in Telemedicine: Compressed ECG signals can be transmitted over networks with limited bandwidth, allowing real-time monitoring and remote diagnostics in telemedicine systems.

Wearable Devices: Efficient ECG compression is vital for battery-powered wearable devices that continuously monitor heart activity, as it reduces the data that needs to be transmitted or stored locally.

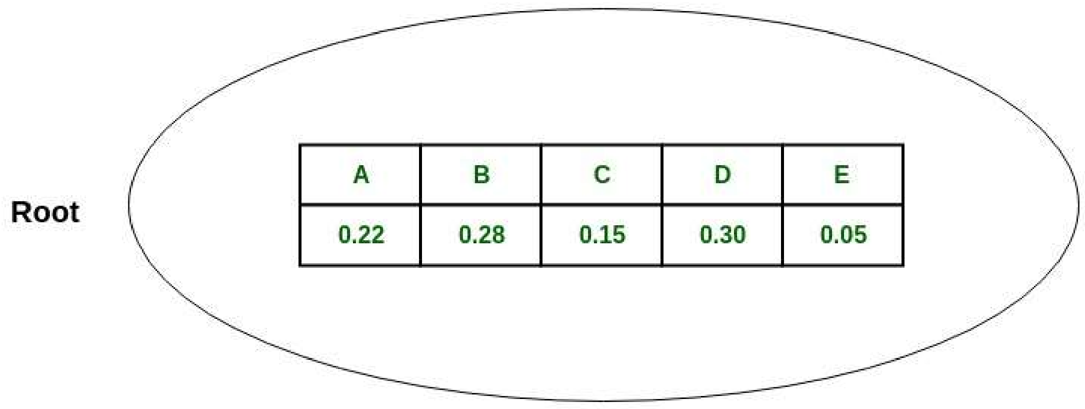
CHAPTER-5

DESIGN & MATHEMATICAL MODEL

SHANNON-FANO TREE

SYMBOL	A	B	C	D	E
PROBABILITY OR FRQUENCY	0.22	0.28	0.15	0.30	0.05

THE SYMBOLS AND THEIR PROBABILITY / FREQUENCY
ARE TAKEN AS INPUTS.
(In case of Frequency, the values can be any number)

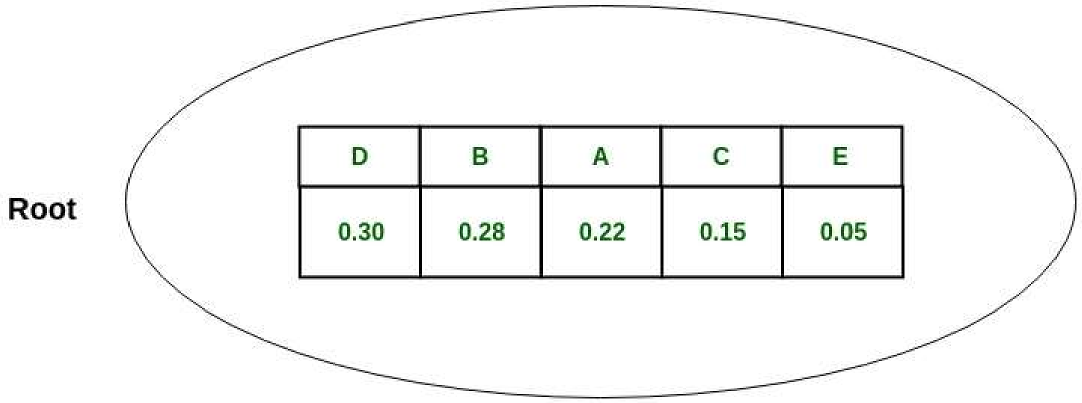


A	B	C	D	E
0.22	0.28	0.15	0.30	0.05

TREE AFTER STEP 1

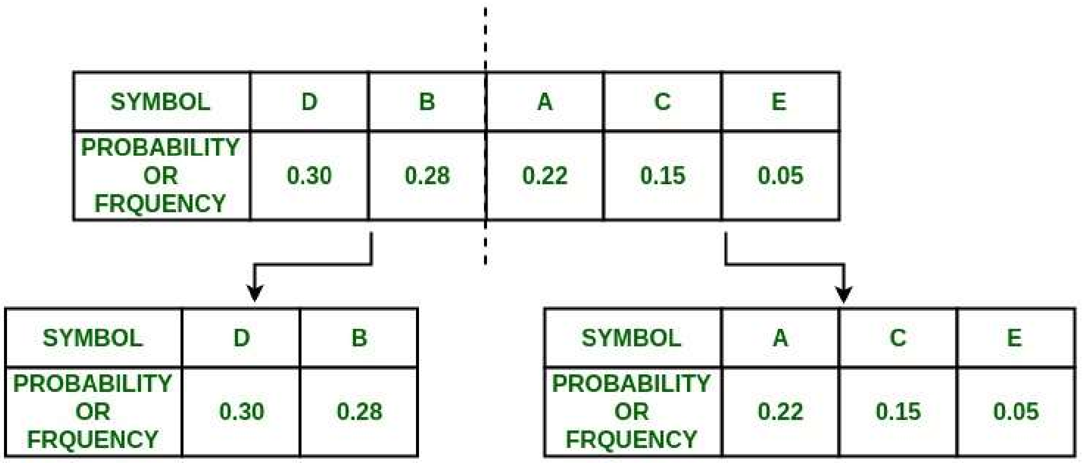
SYMBOL	D	B	A	C	E
PROBABILITY OR FRQUENCY	0.30	0.28	0.22	0.15	0.05

INPUTS ARE SORTED ACCORDING TO THEIR PROBABILITY / FREQUENCY
(Here they are sorted according to their probability)



D	B	A	C	E
0.30	0.28	0.22	0.15	0.05

TREE AFTER STEP 2

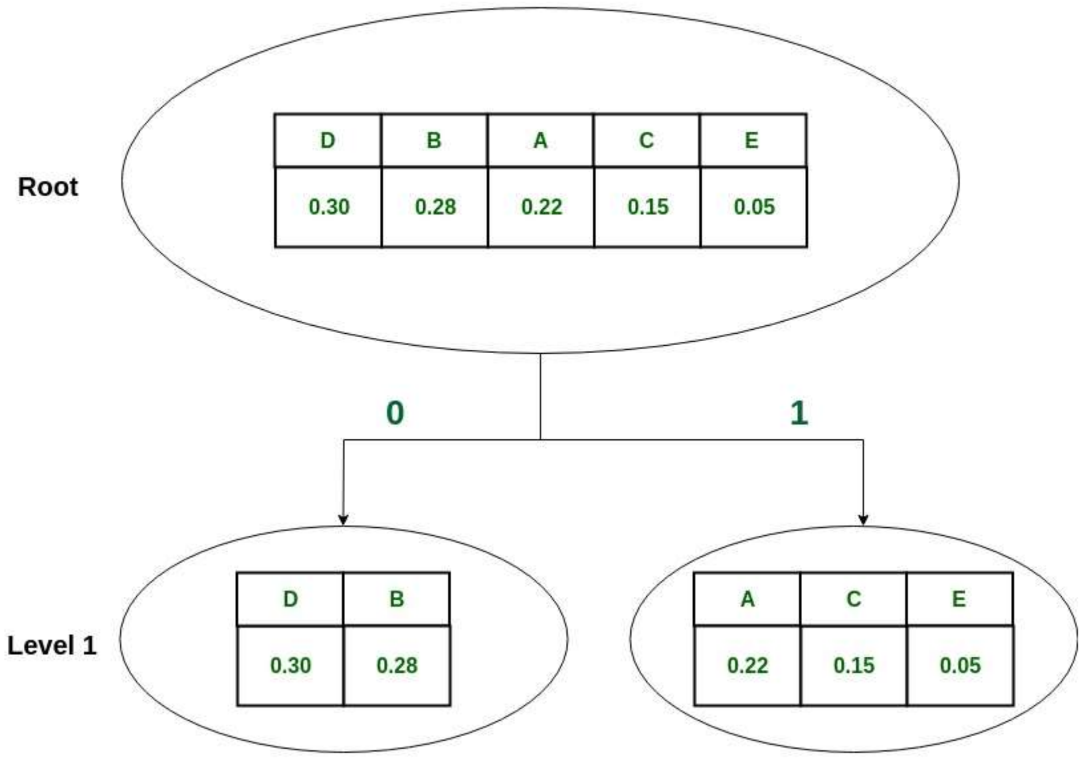


SYMBOL	D	B	A	C	E
PROBABILITY OR FRQUENCY	0.30	0.28	0.22	0.15	0.05

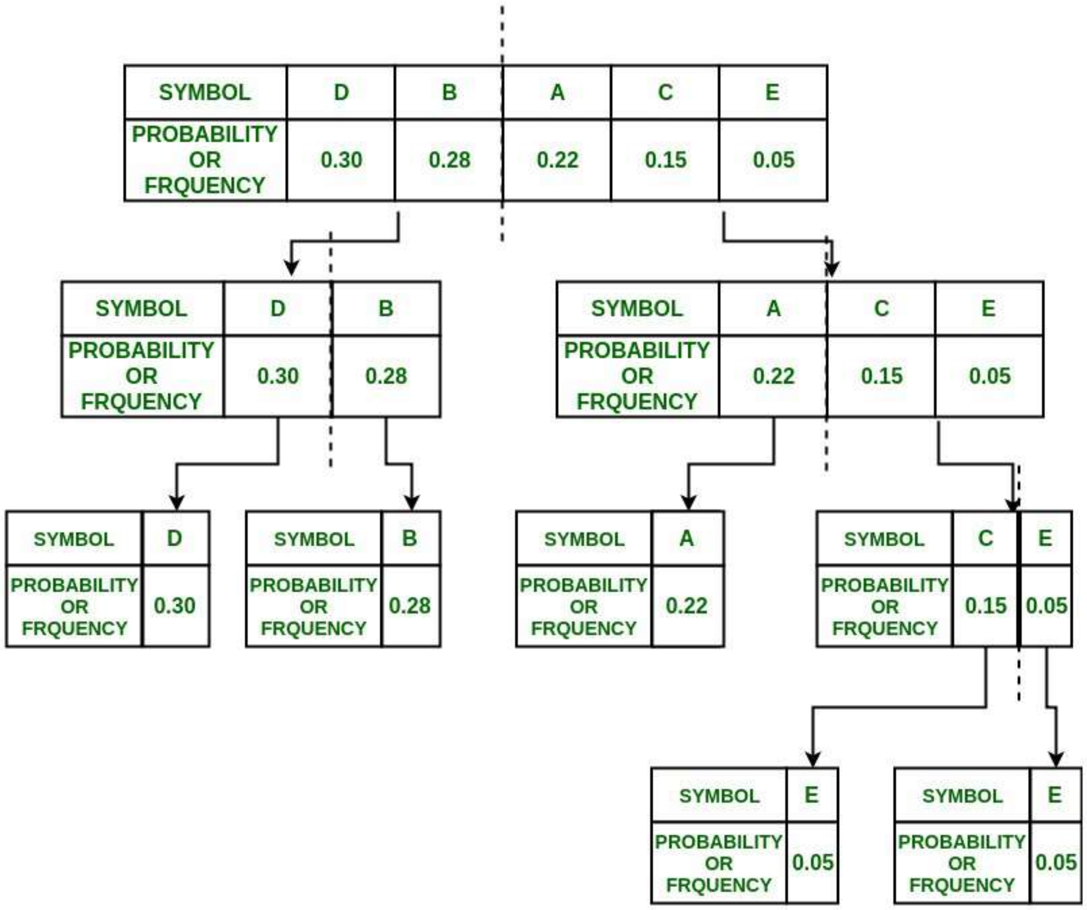
SYMBOL	D	B
PROBABILITY OR FRQUENCY	0.30	0.28

SYMBOL	A	C	E
PROBABILITY OR FRQUENCY	0.22	0.15	0.05

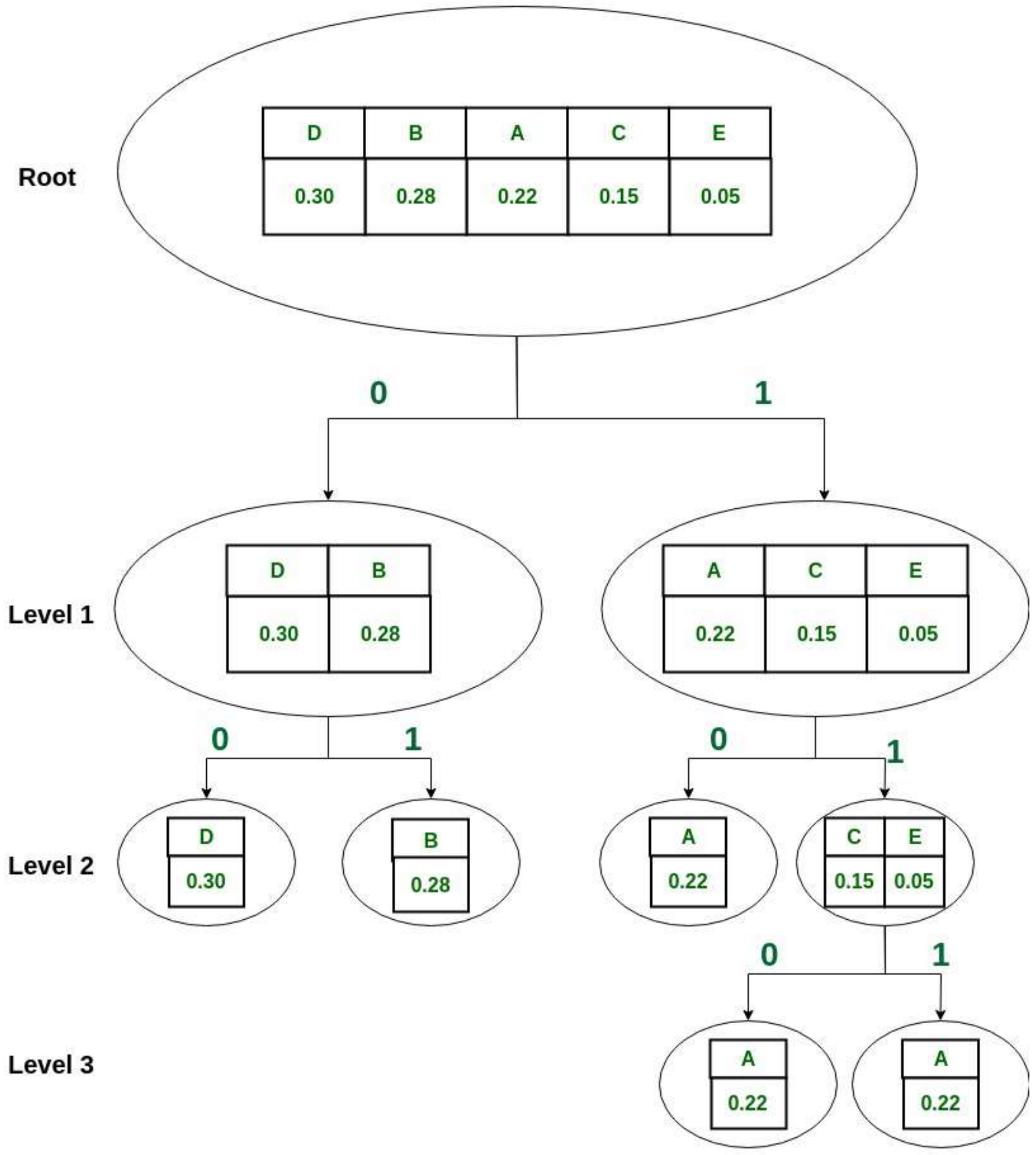
THE SYMBOLS ARE DIVIDED INTO TWO SUCH THAT THE TOTAL PROBABILITY / FREQUENCY OF LEFT SIDE ALMOST SAME AS THAT OF RIGHT SIDE



TREE AFTER STEP 3

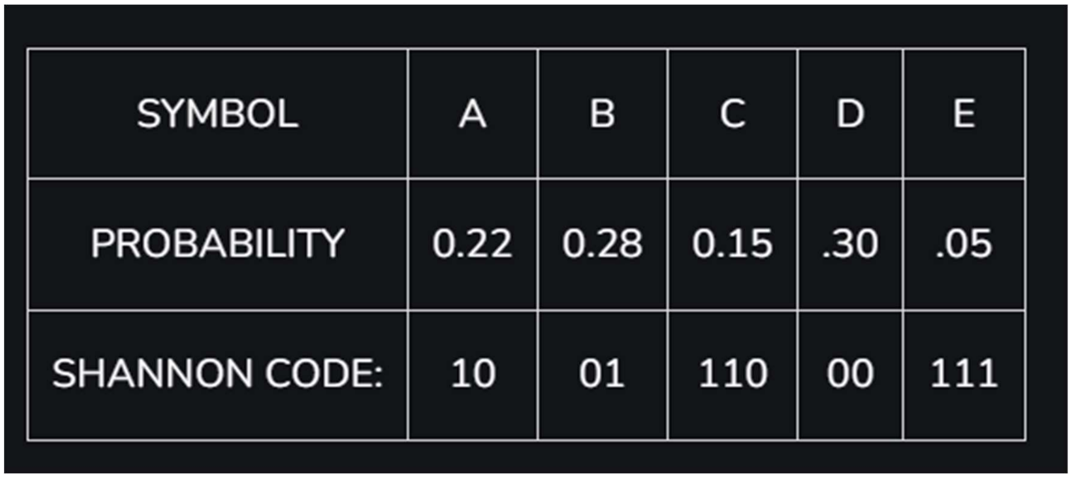

SYMBOL D B A C E
PROBABILITY OR FRQUENCY 0.30 0.28 0.22 0.15 0.05
SYMBOL D B
PROBABILITY OR FRQUENCY 0.30 0.28
SYMBOL A C E
PROBABILITY OR FRQUENCY 0.22 0.15 0.05
SYMBOL D
PROBABILITY OR FRQUENCY 0.30
SYMBOL B
PROBABILITY OR FRQUENCY 0.28
SYMBOL A
PROBABILITY OR FRQUENCY 0.22
SYMBOL C E
PROBABILITY OR FRQUENCY 0.15 0.05
SYMBOL E
PROBABILITY OR FRQUENCY 0.05
SYMBOL E
PROBABILITY OR FRQUENCY 0.05

THE SYMBOLS ARE CONTINUED TO BE DIVIDED INTO TWO
TILL EACH SYMBOL BECOME SEPARATED



TREE AFTER STEP 4

The correct table should be:



SYMBOL	A	B	C	D	E
PROBABILITY	0.22	0.28	0.15	.30	.05
SHANNON CODE:	10	01	110	00	111

MATHEMATICAL MODEL

1. The average length of the Shannon-fano code is:

$$\text{Lavg} = \sum_{i=0}^{k} \text{Pi Li}$$

Where: Pi = Probability

Li = Length of the i^{th} code word

2. Efficiency of the Shannon-Fano code is:

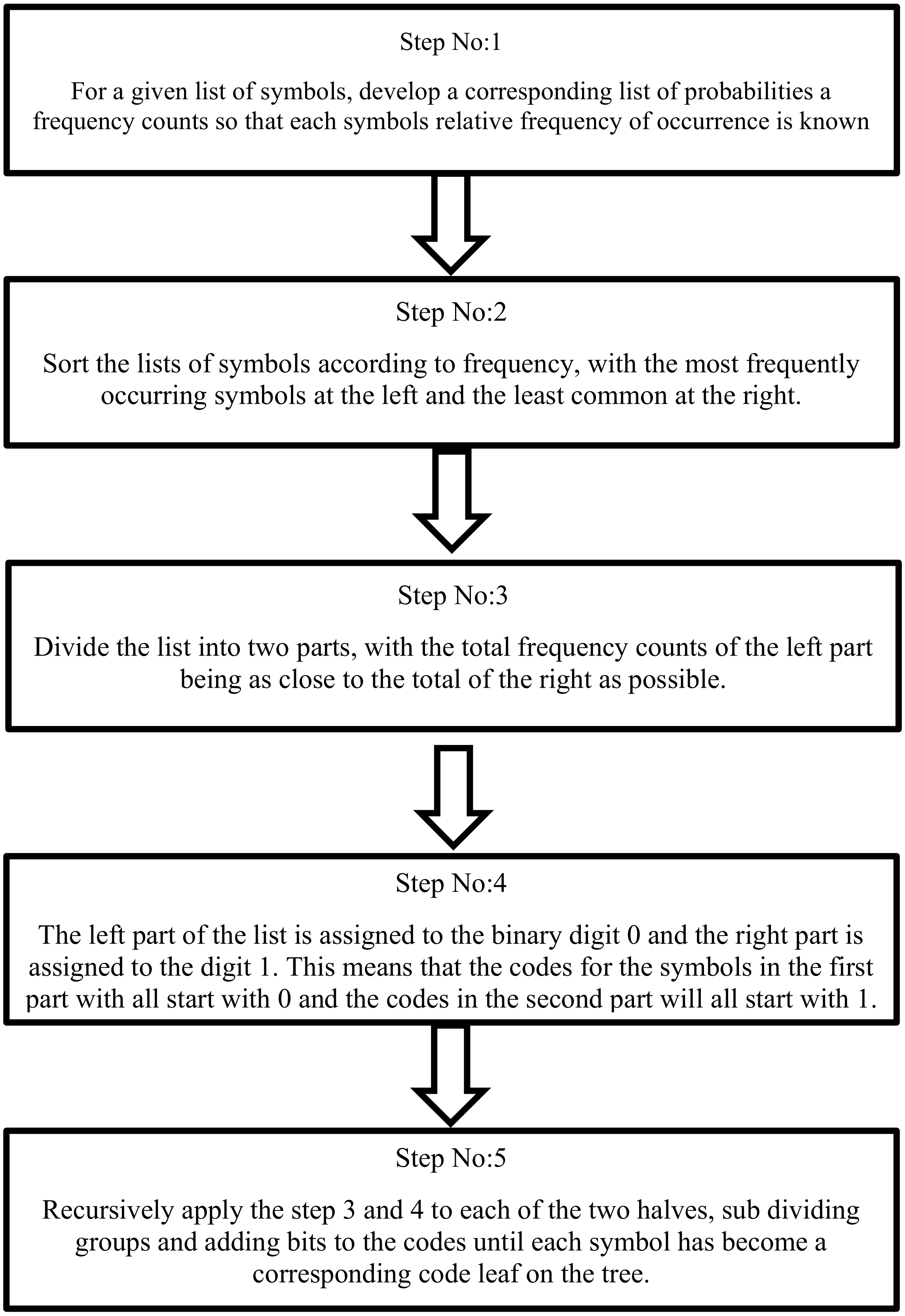
$$\eta = \frac{Entropy\ (H)}{Average\ code\ word\ lengt\ \ (Lavg)}$$

Where H $=\sum_{i=1}^{k} Pi\ log_2\ \frac{1}{Pi}$

3. Variance, $\sigma^2 = \sum_{i=0}^{k}$ [Pi $(Li - \ Lav)^2$]

CHAPTER-6

BLOCK DIAGRAM / FLOW CHART



Step No:1

For a given list of symbols, develop a corresponding list of probabilities a frequency counts so that each symbols relative frequency of occurrence is known

Step No:2

Sort the lists of symbols according to frequency, with the most frequently occurring symbols at the left and the least common at the right.

Step No:3

Divide the list into two parts, with the total frequency counts of the left part being as close to the total of the right as possible.

Step No:4

The left part of the list is assigned to the binary digit 0 and the right part is assigned to the digit 1. This means that the codes for the symbols in the first part with all start with 0 and the codes in the second part will all start with 1.

Step No:5

Recursively apply the step 3 and 4 to each of the two halves, sub dividing groups and adding bits to the codes until each symbol has become a corresponding code leaf on the tree.

CHAPTER-7

PYTHON CODE

Shanon-Fano code:

```
import pandas as pd

import numpy as np

from collections import Counter


# Function to implement the Shannon-Fano algorithm

def shannon_fano(data, prefix=""):

    if len(data) == 1:

        return {data[0][0]: prefix}


    total_weight = sum([item[1] for item in data])

    split_point = 0

    for i in range(len(data)):

        if sum([item[1] for item in data[:i + 1]]) >= total_weight / 2:

            split_point = i + 1

            break


    left = shannon_fano(data[:split_point], prefix + "0")

    right = shannon_fano(data[split_point:], prefix + "1")


    left.update(right)

    return left


# Load data from an Excel sheet

def load_data_from_excel(file_path, sheet_name):

    df = pd.read_excel(file_path, sheet_name=sheet_name)
    values = df.values.flatten().tolist()  # Convert to a flat list
    return values


# Main function to apply Shannon-Fano coding and calculate metrics
def apply_shannon_fano(file_path, sheet_name):
    values = load_data_from_excel(file_path, sheet_name)


    # Calculate the frequency of each unique value
    frequency_dict = Counter(values)
    data = list(frequency_dict.items())  # Convert to list of (value, frequency) pairs


    # Sort data by frequency in descending order
    data.sort(key=lambda x: x[1], reverse=True)


    # Generate Shannon-Fano code
    shannon_fano_code = shannon_fano(data)


    # Convert the data and the corresponding code into a DataFrame
    coded_data = pd.DataFrame(list(shannon_fano_code.items()), columns=["Value", "Shannon-
Fano Code"])


    # Create frequency DataFrame
    freq_df = pd.DataFrame(data, columns=['Value', 'Frequency'])


    # Merge coded_data and freq_df
    merged_data = pd.merge(coded_data, freq_df, on='Value')


    # Compute total frequency
    total_frequency = merged_data['Frequency'].sum()
    # Compute probabilities
    merged_data['Probability'] = merged_data['Frequency'] / total_frequency

    # Compute codeword lengths
    merged_data['Codeword Length'] = merged_data['Shannon-Fano Code'].apply(len)

    # Compute average codeword length
    avg_codeword_length = (merged_data['Probability'] * merged_data['Codeword
Length']).sum()

    # Compute entropy
    merged_data['Entropy Term'] = merged_data['Probability'] *
np.log2(merged_data['Probability'])
    entropy = -merged_data['Entropy Term'].sum()

    # Compute efficiency
    efficiency = entropy / avg_codeword_length

    # Compute variance of codeword lengths
    merged_data['Variance Term'] = merged_data['Probability'] * (merged_data['Codeword
Length'] - avg_codeword_length) ** 2
    variance = merged_data['Variance Term'].sum()

    # Print the calculated values
    print(f"Average Codeword Length: {avg_codeword_length}")
    print(f"Entropy: {entropy}")
    print(f"Efficiency: {efficiency}")
    print(f"Variance of Codeword Lengths: {variance}")

    # Return the merged data and metrics
    return merged_data[['Value', 'Frequency', 'Probability', 'Shannon-Fano Code', 'Codeword
Length']], avg_codeword_length, entropy, efficiency, variance


# Save data to CSV, including the calculated metrics

def save_to_csv(file_path, shannon_fano_data, avg_length, entropy, efficiency, variance):

    # Save the Shannon-Fano coded data to a CSV file

    shannon_fano_data.to_csv(file_path + '_shannon_fano_coded_data.csv', index=False)


    # Create a DataFrame for the metrics

    metrics_df = pd.DataFrame({

        'Metric': ['Average Codeword Length', 'Entropy', 'Efficiency', 'Variance'],

        'Value': [avg_length, entropy, efficiency, variance]

    })


    # Save the metrics to another CSV file

    metrics_df.to_csv(file_path + '_shannon_fano_metrics.csv', index=False)


# Main code

file_path = r"C:\shannon fano project\data20"

sheet_name = 'Sheet1'


# Apply Shannon-Fano coding and get metrics

shannon_fano_data, avg_length, entropy, efficiency, variance = apply_shannon_fano(file_path +
".xlsx", sheet_name)


# Save the data and metrics to CSV

save_to_csv(file_path, shannon_fano_data, avg_length, entropy, efficiency, variance)


# Print some of the coded data and metrics for reference

print(shannon_fano_data.head())  # First few rows of the Shannon-Fano coded data
```

Graphical parameters code:

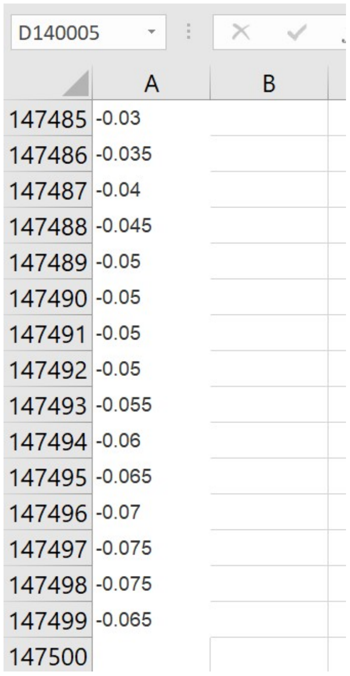
```
import pandas as pd

import matplotlib.pyplot as plt


# Function to load the data from the Excel sheet

def load_metrics_from_excel(file_path, sheet_name):

    # Load data from the Excel file

    df = pd.read_excel(file_path, sheet_name=sheet_name)


    # Assuming the metrics are in the first few columns

    # Adjust these columns depending on where your metrics are stored

    sample_sizes = df.iloc[1:21, 0].values  # First column: Sample size (rows 2 to 16)

    avg_codeword_lengths = df.iloc[1:21, 1].values  # Second column: Avg Codeword Length

    entropies = df.iloc[1:21, 2].values  # Third column: Entropy

    efficiencies = df.iloc[1:21, 3].values  # Fourth column: Efficiency

    variances = df.iloc[1:21, 4].values  # Fifth column: Variance


    return sample_sizes, avg_codeword_lengths, entropies, efficiencies, variances


# Function to plot the metrics

def plot_metrics(sample_sizes, avg_codeword_lengths, entropies, efficiencies, variances):

    plt.figure(figsize=(10, 6))


    # Plot Average Codeword Length

    plt.subplot(2, 2, 1)

    plt.plot(sample_sizes, avg_codeword_lengths, marker='o', label="Avg Codeword Length")

    plt.xlabel('Sample Size')

    plt.ylabel('Avg Codeword Length')

    plt.title('Average Codeword Length vs Sample Size')
    # Plot Entropy

    plt.subplot(2, 2, 2)

    plt.plot(sample_sizes, entropies, marker='o', label="Entropy")

    plt.xlabel('Sample Size')

    plt.ylabel('Entropy')

    plt.title('Entropy vs Sample Size')


    # Plot Efficiency

    plt.subplot(2, 2, 3)

    plt.plot(sample_sizes, efficiencies, marker='o', label="Efficiency")

    plt.xlabel('Sample Size')

    plt.ylabel('Efficiency')

    plt.title('Efficiency vs Sample Size')


    # Plot Variance

    plt.subplot(2, 2, 4)

    plt.plot(sample_sizes, variances, marker='o', label="Variance")

    plt.xlabel('Sample Size')

    plt.ylabel('Variance')

    plt.title('Variance vs Sample Size')


    plt.tight_layout()

    plt.show()


# File path and sheet name details

file_path = r"C:\shannon fano project\overall_shannon_fano_metrics.xlsx"

sheet_name = 'overall_shannon_fano_metrics'
```

```
# Load the metrics data

sample_sizes, avg_codeword_lengths, entropies, efficiencies, variances =
load_metrics_from_excel(file_path, sheet_name)


# Plot the metrics

plot_metrics(sample_sizes, avg_codeword_lengths, entropies, efficiencies, variances)
```

CHAPTER-8

RESULT WITH GRAPH

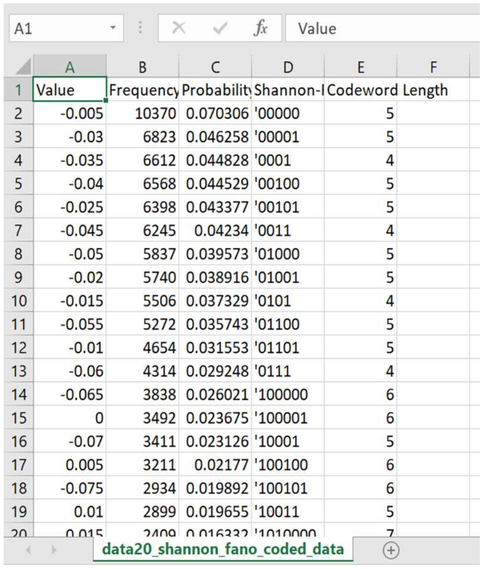
Sampled and filtered data sets of 20 people (147499 samples)



D140005

	A	B
147485	-0.03	
147486	-0.035	
147487	-0.04	
147488	-0.045	
147489	-0.05	
147490	-0.05	
147491	-0.05	
147492	-0.05	
147493	-0.055	
147494	-0.06	
147495	-0.065	
147496	-0.07	
147497	-0.075	
147498	-0.075	
147499	-0.065	
147500		

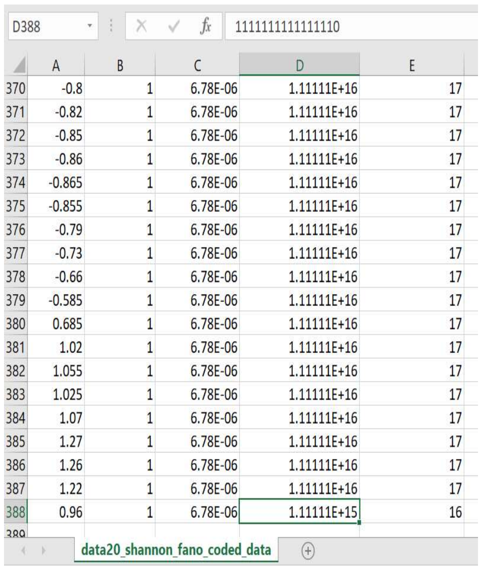
Shannon-Fano coded data (388 code words)



A1 Value

	A	B	C	D	E	F
1	Value	Frequency	Probability	Shannon-I	Codeword	Length
2	-0.005	10370	0.070306	'00000	5	
3	-0.03	6823	0.046258	'00001	5	
4	-0.035	6612	0.044828	'0001	4	
5	-0.04	6568	0.044529	'00100	5	
6	-0.025	6398	0.043377	'00101	5	
7	-0.045	6245	0.04234	'0011	4	
8	-0.05	5837	0.039573	'01000	5	
9	-0.02	5740	0.038916	'01001	5	
10	-0.015	5506	0.037329	'0101	4	
11	-0.055	5272	0.035743	'01100	5	
12	-0.01	4654	0.031553	'01101	5	
13	-0.06	4314	0.029248	'0111	4	
14	-0.065	3838	0.026021	'100000	6	
15	0	3492	0.023675	'100001	6	
16	-0.07	3411	0.023126	'10001	5	
17	0.005	3211	0.02177	'100100	6	
18	-0.075	2934	0.019892	'100101	6	
19	0.01	2899	0.019655	'10011	5	
20	0.015	2409	0.016332	'1010000	7	

data20_shannon_fano_coded_data

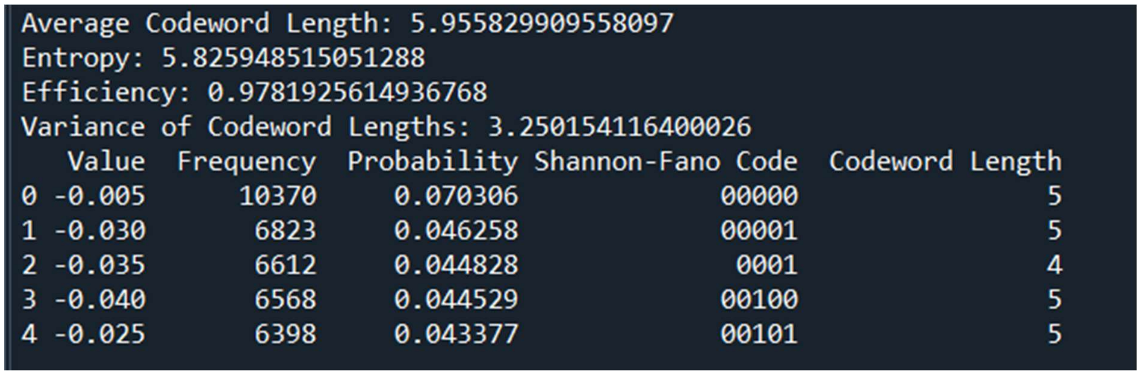


D388 1111111111111110

	A	B	C	D	E
370	-0.8	1	6.78E-06	1.11111E+16	17
371	-0.82	1	6.78E-06	1.11111E+16	17
372	-0.85	1	6.78E-06	1.11111E+16	17
373	-0.86	1	6.78E-06	1.11111E+16	17
374	-0.865	1	6.78E-06	1.11111E+16	17
375	-0.855	1	6.78E-06	1.11111E+16	17
376	-0.79	1	6.78E-06	1.11111E+16	17
377	-0.73	1	6.78E-06	1.11111E+16	17
378	-0.66	1	6.78E-06	1.11111E+16	17
379	-0.585	1	6.78E-06	1.11111E+16	17
380	0.685	1	6.78E-06	1.11111E+16	17
381	1.02	1	6.78E-06	1.11111E+16	17
382	1.055	1	6.78E-06	1.11111E+16	17
383	1.025	1	6.78E-06	1.11111E+16	17
384	1.07	1	6.78E-06	1.11111E+16	17
385	1.27	1	6.78E-06	1.11111E+16	17
386	1.26	1	6.78E-06	1.11111E+16	17
387	1.22	1	6.78E-06	1.11111E+16	17
388	0.96	1	6.78E-06	1.11111E+15	16
389					

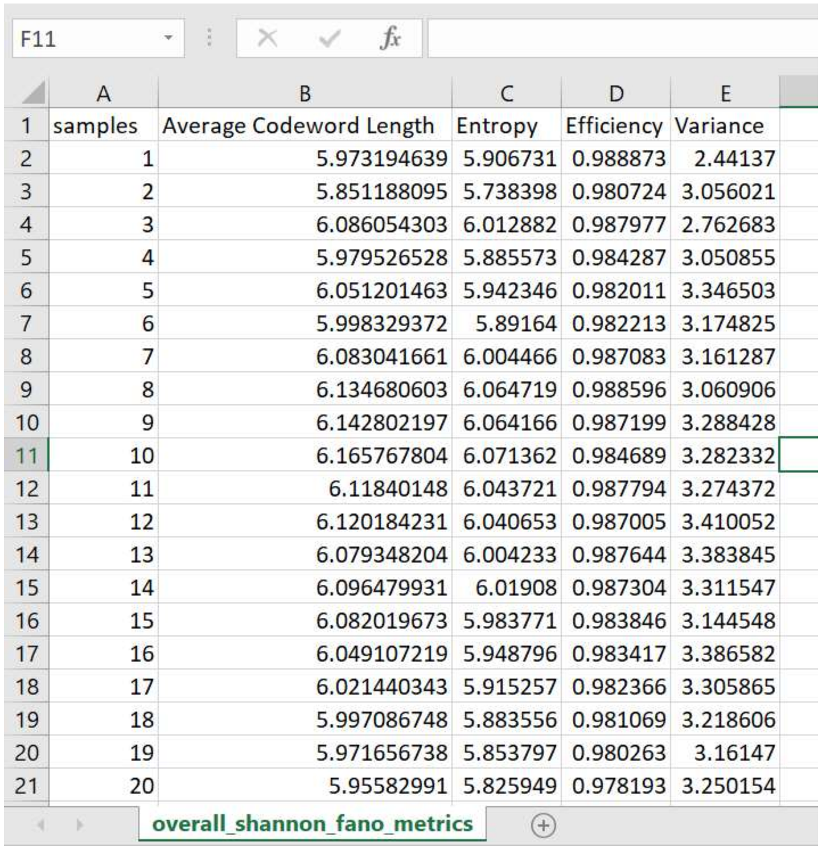
data20_shannon_fano_coded_data

Shannon-Fano Metrics (20 People Dataset):



```
Average Codeword Length: 5.955829909558097
Entropy: 5.825948515051288
Efficiency: 0.9781925614936768
Variance of Codeword Lengths: 3.250154116400026
   Value  Frequency  Probability Shannon-Fano Code  Codeword Length
0 -0.005      10370     0.070306             00000                5
1 -0.030       6823     0.046258             00001                5
2 -0.035       6612     0.044828              0001                4
3 -0.040       6568     0.044529             00100                5
4 -0.025       6398     0.043377             00101                5
```

Shannon Fano Metrics with Varying Sample Size

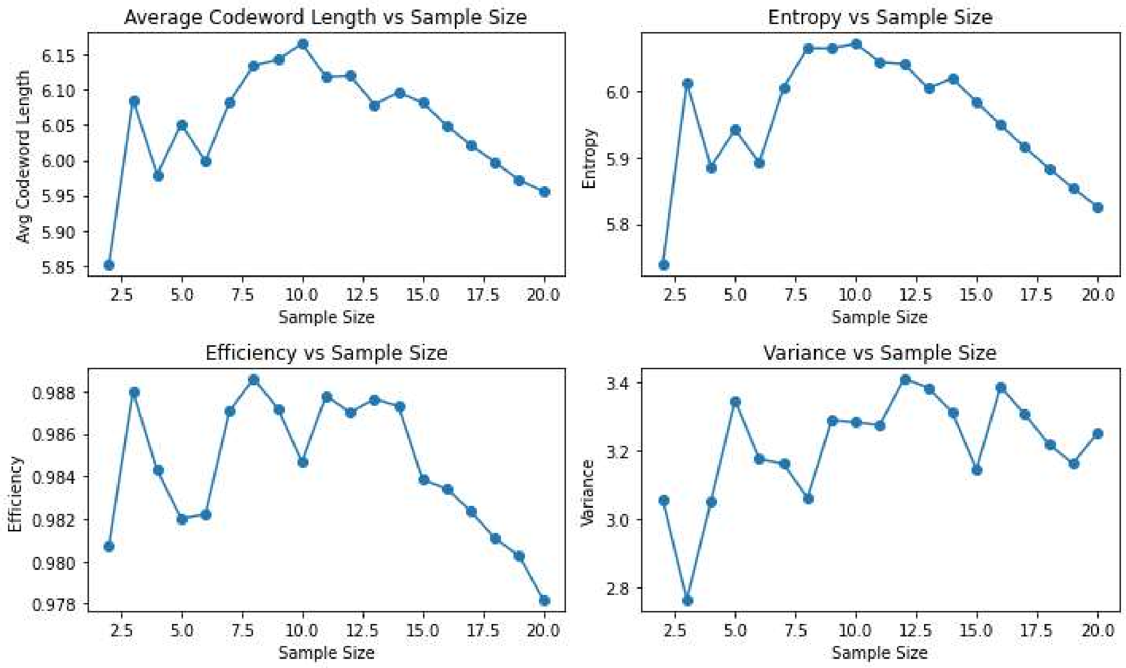


F11

	A	B	C	D	E
1	samples	Average Codeword Length	Entropy	Efficiency	Variance
2	1	5.973194639	5.906731	0.988873	2.44137
3	2	5.851188095	5.738398	0.980724	3.056021
4	3	6.086054303	6.012882	0.987977	2.762683
5	4	5.979526528	5.885573	0.984287	3.050855
6	5	6.051201463	5.942346	0.982011	3.346503
7	6	5.998329372	5.89164	0.982213	3.174825
8	7	6.083041661	6.004466	0.987083	3.161287
9	8	6.134680603	6.064719	0.988596	3.060906
10	9	6.142802197	6.064166	0.987199	3.288428
11	10	6.165767804	6.071362	0.984689	3.282332
12	11	6.11840148	6.043721	0.987794	3.274372
13	12	6.120184231	6.040653	0.987005	3.410052
14	13	6.079348204	6.004233	0.987644	3.383845
15	14	6.096479931	6.01908	0.987304	3.311547
16	15	6.082019673	5.983771	0.983846	3.144548
17	16	6.049107219	5.948796	0.983417	3.386582
18	17	6.021440343	5.915257	0.982366	3.305865
19	18	5.997086748	5.883556	0.981069	3.218606
20	19	5.971656738	5.853797	0.980263	3.16147
21	20	5.95582991	5.825949	0.978193	3.250154

overall_shannon_fano_metrics

Graphical Plot:



CHAPTER 9

CONCLUSION

In this project, we successfully implemented Shannon-Fano coding to compress ECG signals from a dataset comprising 147,499 samples collected from 20 individuals. The coding process generated 388 unique code words, leading to notable metrics that highlight the efficiency and effectiveness of the compression. The average codeword length was approximately 5.96 bits, while the entropy of the encoded data was calculated to be around 5.83 bits, indicating a high degree of information representation with minimal redundancy. The achieved efficiency of 0.98 underscores the effectiveness of the Shannon-Fano coding technique in this context, and the variance in codeword lengths, measured at approximately 3.25, reflects the diversity in signal characteristics across different ECG readings.

Looking ahead, potential improvements could include exploring adaptive coding techniques or integrating machine learning algorithms to enhance the compression process further. Additionally, extending the dataset to include a wider variety of ECG signals from diverse demographics could improve the model's robustness.

The applications of this work in healthcare are significant, particularly in telemedicine and remote patient monitoring systems. Efficient compression of ECG signals can facilitate faster transmission of vital health data over limited bandwidths, enabling timely diagnosis and monitoring of patients' cardiac health. Ultimately, advancements in data compression techniques like Shannon-Fano coding will contribute to the development of more efficient and effective healthcare technologies, improving patient outcomes and optimizing resource utilization.

CHAPTER 10

REFERENCES

- *Communication Systems, Simon Haykin, Wiley, Fourth Edition.*

- *Oppenheim, A. V., & Willsky, A. S. (1997). Signals and Systems. 2nd Edition. Prentice Hall.*

- LW: .../1.0.0/Person_01/rec_1 (physionet.org)