Assignment Report

BS CIS(24-28)

Semester: 03

Linear Algebra

Sir Zaheer Asghar

20/12/25

Group Members:

1. Aameen Fatima
2. Ayesha Omar
3. Hadia Sohail
4. Hafsa Naz
5. Meshal Gul

Abstract

The rapid development of quantum computing poses a serious threat to classical public-key cryptographic systems such as RSA and ECC, which form the backbone of today's digital security. To address this challenge, Post-Quantum Cryptography (PQC) has emerged as a new class of cryptographic algorithms designed to remain secure against both classical and quantum attacks. In this project, we studied the fundamental principles of PQC and explored how linear algebra plays a central role in both the construction and analysis of post-quantum algorithms.

As part of our Linear Algebra course, we focused on **Principal Component Analysis (PCA)** as a practical application of vectors, matrices, covariance, eigenvalues, and eigenvectors. We applied PCA to a real dataset consisting of performance metrics from 25 standardized PQC algorithms, including key generation time, encryption and decryption time, ciphertext size, and memory overhead. Using linear algebra techniques, we transformed the high-dimensional data into a lower-dimensional space while preserving the most significant patterns.

Our results show that the first two principal components capture more than 78% of the total variance, allowing clear visualization and comparison of cryptographic algorithms. PCA revealed meaningful clusters among algorithm families and highlighted important trade-offs between performance, memory usage, and security levels. This demonstrates how abstract linear algebra concepts such as eigen decomposition and vector projections can be directly applied to analyze real-world cybersecurity problems.

Overall, this project strengthened our understanding of PCA from both a theoretical and practical perspective and highlighted the growing importance of linear algebra in modern cryptography and data analysis, especially in the post-quantum era.

POST QUANTUM CRYPTOGRAPHY

1.Post Quantum Cryptography

1.1 What Is Post-Quantum Cryptography (PQC)?

Post-Quantum Cryptography (PQC) refers to cryptographic algorithms that are designed to remain secure even if large-scale quantum computers are built in the future. These algorithms aim to protect data and communications against attacks by both classical and quantum computers. PQC focuses on replacing current public-key systems that are vulnerable to quantum algorithms with new mathematical problems believed to be hard for any computer to solve efficiently.

1.2 Why Do We Need PQC?

Modern encryption, such as RSA and elliptic-curve cryptography (ECC), depends on mathematical problems that are extremely hard for classical computers to solve. However, *quantum computers* can solve these problems much faster using quantum algorithms, meaning today's encryption could be broken in the future. This vulnerability threatens the security of internet traffic, digital identities, and financial systems protected by current cryptographic methods.

1.3 What Are Quantum Computers?

Quantum computers use quantum bits called **qubits**, which can represent both 0 and 1 at the same time due to principles of quantum superposition. This allows them to perform certain calculations *much more efficiently* than classical computers, which process bits in only one state at a time. Because of this, quantum computers can potentially solve complex mathematical problems more quickly than classical systems, posing a threat to cryptographic schemes that rely on such hard problems.

1.4 Classical Algorithms and Their Vulnerability to Quantum Attacks

Classical public-key cryptography systems such as **RSA, Diffie–Hellman, and ECC** protect data by relying on problems like integer factorization and discrete logarithms, which are computationally difficult for current computers to solve. For example:

- **RSA** depends on factoring large numbers into primes.
- **Diffie–Hellman** depends on discrete logarithms in groups.

Quantum algorithms—especially **Shor's algorithm**—can solve these hard problems *efficiently* on a sufficiently powerful quantum computer. This means that once such quantum computers exist, they could break RSA and Diffie–Hellman encryption by solving the underlying math quickly, making encrypted messages readable by attackers.

1.5 How Are PQC Algorithms Different from Classical Ones?

Unlike classical public-key systems that are vulnerable to quantum attacks, post-quantum algorithms are designed to resist both classical and quantum computational attacks. They are *not* quantum algorithms; rather, they **run on the same classical hardware** we use today but are based on different mathematical problems that are believed to be hard for quantum computers to solve.

https://www.paloaltonetworks.com/cyberpedia/what-is-post-quantum-cryptography-pqc

Aspect	Classical public-key cryptography	Post-quantum cryptography (PQC)
Mathematical foundation	Based on factoring (RSA) and discrete logarithms (ECC, Diffie–Hellman)	Based on problems like lattices, hash chains, error-correcting codes, and multivariate equations
Vulnerability to quantum attacks	Breakable by Shor's algorithm on large-scale quantum computers	Designed to resist both classical and quantum attacks
Hardware requirements	Runs on classical computers	Runs on classical computers — no quantum hardware needed
Security assumption	Computational difficulty for classical systems	Hardness believed to hold against classical and quantum attacks
Examples	RSA, ECDSA, Diffie–Hellman	ML-KEM (KEM), ML-DSA (signatures), SLH-DSA (signatures), Classic McEliece (KEM).
Readiness for deployment	Already in use globally but becoming obsolete under quantum threat	NIST standards finalized and ready for phased deployment

1.6 Types of Post-Quantum Cryptography Algorithms

PQC is not a single algorithm, but an ecosystem of algorithms based on mathematical problems that remain hard for both classical and quantum computers. Major families include:

1.6.1. Lattice-Based Cryptography

Lattice-based algorithms rely on mathematical problems in high-dimensional lattices. They are considered strong candidates for post-quantum security because the underlying problems (like Learning With Errors and Short Integer Solution) are believed to be hard even for quantum computers. Examples include key-encapsulation mechanisms (KEMs) like **ML-KEM**.

1.6.2. Hash-Based Cryptography

These methods use one-way hash functions to create digital signatures. They are considered quantum-safe because reversing hash functions is difficult for both classical and quantum systems. Algorithms like **SLH-DSA** are examples.

1.6.3. Code-Based Cryptography

Code-based systems rely on the difficulty of decoding random linear codes. Classic systems like **McEliece** have stood up to decades of cryptanalysis and are believed to resist quantum attacks, though they often require large key sizes.

1.6.4. Multivariate Cryptography

These are based on solving systems of nonlinear polynomial equations, which are believed to be hard for quantum and classical computers. They are mainly used for digital signatures.

Source:

Palo Alto Networks

Here are the details of these algorithms which shows their connection with linear algebra:

1.7 Linear Algebra and Post Quantum Cryptography Algorithms

1.7.1 Lattice-Based Cryptography

Lattice-based cryptography is a class of cryptographic systems whose security relies on the computational hardness of certain problems defined on mathematical lattices. A lattice is a discrete set of points in an n-dimensional space formed by all integer linear combinations of a set of basis vectors. Due to their strong resistance against both classical and quantum attacks, lattice-based schemes are considered strong candidates for post-quantum cryptography.

One of the earliest lattice-based public-key cryptosystems is the **Goldreich–Goldwasser–Halevi (GGH) cryptosystem**, which demonstrates how lattice problems can be effectively used for encryption and decryption.

Mathematical Definition of a Lattice

In linear algebra, a lattice (L) generated by basis vectors

$$\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n$$

is defined as:

$$L = \left\{ \sum_{i=1}^{n} z_i \, \mathbf{b}_i \mid z_i \in \mathbb{Z} \right\}$$

This definition shows that a lattice is formed by **integer linear combinations of vectors**, directly linking lattice cryptography with vector spaces and matrix representations in linear algebra.

GGH Lattice-Based Cryptography Scheme

The GGH cryptosystem is based on the computational difficulty of the **Closest Vector Problem (CVP)** in high-dimensional lattices. The system uses two different bases of the same lattice:

- **Good basis (private key):** orthogonal and easy to compute with
- **Bad basis (public key):** highly non-orthogonal and difficult to use.

The public key is derived from the private key through linear algebraic transformations, making it computationally infeasible to recover the good basis from the bad basis.

Encryption Process in GGH

In the GGH scheme, a plaintext message is represented as a vector in Euclidean space. Encryption is performed through the following steps:

1. Representing the message as a vector
2. Multiplying it with the public lattice basis (matrix multiplication)

3. Adding a small random error vector

Mathematically, encryption is expressed as:

$$\mathbf{c} = \mathbf{B}_{\text{public}}\mathbf{m} + \mathbf{e}$$

where:

- $\mathbf{B}_{\text{public}}$ is the public lattice basis matrix.
- **m** is the message vector.
- **e** is a small error vector.

This process relies heavily on matrix operations and vector addition, which are core topics of linear algebra.

Decryption Process in GGH

Decryption is performed using the private (good) basis of the lattice. Because this basis is nearly orthogonal, it allows efficient computation of the lattice point closest to the received ciphertext vector. This step involves solving a nearest-vector approximation problem, which is computationally easy with the private basis but extremely difficult with the public basis.

Security of GGH and Its Connection to Linear Algebra

The security of the GGH cryptosystem relies on the hardness of fundamental lattice problems such as:

- Closest Vector Problem (CVP)
- Shortest Vector Problem (SVP)

These problems involve essential linear algebra concepts, including:

- Vector norms
- Distance calculations
- Basis transformations
- Matrix representations

Due to this strong reliance on linear algebra, lattice-based cryptography serves as a natural and powerful application of linear algebra in post-quantum security systems.

Linear Algebra Interpretation of Lattice Cryptography

Lattice-based cryptography is deeply rooted in linear algebra, as it relies on vector spaces, matrices, and geometric transformations in high-dimensional spaces. A lattice can be viewed as a discrete subset of an n-dimensional vector space formed by all integer linear combinations of linearly independent basis vectors.

From a linear algebra perspective, a lattice basis can be represented as a matrix:

$$\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots \dots \dots \dots . . \mathbf{b}_n]$$

where each column vector $\mathbf{b}_i$ represents a basis vector of the lattice. Any lattice point can be expressed as:

$$\mathbf{v} = \mathbf{Bz}, \mathbf{z} \in \mathbb{Z}^n$$

This formulation highlights the importance of matrix–vector multiplication in lattice cryptography.

Orthogonality and Numerical Stability

Orthogonality is a key concept in linear algebra that directly affects cryptographic security. A basis with nearly orthogonal vectors allows efficient computation of projections and distances, making decryption possible using the private key.

In contrast, a non-orthogonal basis leads to numerical instability during distance calculations. This instability makes problems such as finding the closest lattice point extremely difficult when only the public key is available.

Vector Norms and Distance Computation

Many lattice problems in cryptography depend on measuring distances between vectors. These distances are computed using vector norms, such as the Euclidean norm:

$$\parallel \mathbf{x} \parallel = \sqrt{\mathbf{x}^T \mathbf{x}}$$

The security of lattice-based cryptography relies on the difficulty of finding vectors with minimum norm or identifying the lattice point closest to a given vector, both of which are geometric problems formulated using linear algebra.

Error Vectors and Perturbation

During encryption, a small error vector is added to the encoded message. From a linear algebra viewpoint, this addition represents a vector perturbation. While this perturbation can be removed using the private basis, it significantly increases the difficulty of decoding using the public basis, thereby enhancing security.

Connection to High-Dimensional Geometry

Lattice cryptography operates in very high-dimensional spaces, often involving hundreds or thousands of dimensions. In such spaces, geometric intuition becomes unreliable, and linear algebra becomes the primary analytical tool. Matrix operations, vector projections, and basis reductions play a central role in understanding and implementing lattice-based cryptographic systems.

Relevance to Post-Quantum Cryptography

Lattice-based cryptographic schemes such as GGH are believed to be resistant to quantum algorithms like Shor’s algorithm. This resistance makes them strong candidates for securing future communication systems against quantum computing threats.

Source

Goldreich, O., Goldwasser, S., & Halevi, S. *GGH Lattice-Based Cryptography*. Scribd. https://www.scribd.com/document/430482830/GGH-Lattice-Based-Cryptography

1.7.2 Hash based Cryptography

Although linear algebra is not central to hash-based cryptography, it appears indirectly in several aspects. Hash inputs and outputs are commonly represented as binary vectors in the vector space $\mathbb{F}_2^n$, which is a linear algebraic structure defined over a finite field [Stinson & Paterson]. Additionally, internal components of hash functions may involve XOR operations and fixed linear transformations, which can be modeled using matrix multiplication over $\mathbb{F}_2$[Daemen & Rijmen]. Furthermore, linear algebra techniques are employed in cryptographic analysis to study diffusion properties and identify linear dependencies during linear cryptanalysis, although these methods serve as analytical tools rather than the security foundation of hash-based cryptography [Katz & Lindell].

1.7.3 Code based Cryptography.

Definition:

Code-based cryptography is a branch of public-key cryptography that uses **error-correcting codes** to secure messages. One of the earliest and most well-known schemes is the **McEliece cryptosystem**, introduced in 1978. It relies on linear error-correcting codes to encode messages, and the security comes from the difficulty of decoding a general linear code without the secret key.

How It Works:

a) A **generator matrix (G)** represents the linear code.

b) The message is treated as a vector $m$and is encoded using matrix multiplication:

$$c = m \cdot G$$

where $c$is the codeword sent to the receiver.

c) The sender adds a small random error to c, making it hard for attackers to decode without the secret key.

d) The receiver, knowing the secret code structure, can efficiently decode $c$to recover m.

Linear Algebra Connection:

- Messages and codewords are represented as **vectors** over a finite field ($\mathbb{F}_2$ for binary codes).
- Encoding and scrambling operations involve **matrix-vector multiplication** and **linear transformations**.
- The cryptographic security relies on the **hardness of solving linear systems with errors**, which is a linear algebra problem over finite fields.

Sources:

- Classic McEliece – Code-based Cryptography
- A Survey on Code-Based Cryptography

1.7.4 Multivariate Cryptography

Definition:

Multivariate cryptography (MQ cryptography) is a type of post-quantum public-key cryptography based on the difficulty of solving systems of **multivariate quadratic equations** over finite fields. Examples include schemes like **Rainbow** and **HFE (Hidden Field Equations)**.

How It Works :

a) The public key is a system of quadratic polynomials in several variables.
b) The private key consists of simple maps and transformations that allow easy inversion.
c) Encryption involves evaluating the public quadratic system; decryption uses the private transformations to recover the message.

Linear Algebra Connection:

- The secret maps often include **invertible linear or affine transformations**, which are **matrix operations on vectors**.
- Solving the public system without the private key is equivalent to solving a system of nonlinear equations. One common attack method is **linearization,** where quadratic terms are treated as new variables, reducing the problem to a **linear algebra system** in a higher-dimensional space.
- Variables, polynomial terms, and linear transformations are all represented using **vectors and matrices over finite fields**, linking multivariate cryptography to linear algebra concepts.

Sources:

- Multivariate Cryptography – Wikipedia
- Intro to Multivariate Cryptography (PDF)

1.8 Major Families of Post-Quantum Cryptographic Algorithms

Algorithm Family	Security Basis	Primary Use	Key Characteristics

Lattice-Based	Hardness of problems in highdimensional geometric lattices (e.g., Learning With Errors - LWE).	Key Establishment & Digital Signatures	Favored for balance of security and efficiency; supports advanced functions like homomorphic encryption.
Hash-Based	Cryptographic strength of one-way hash functions (e.g., SHA-256).	Digital Signatures Only	Considered the most conservative choice due to decades of hash function analysis; simple and mature.
Code-Based	Difficulty of decoding random linear errorcorrecting codes.	Key Establishment	Has a long, unbroken security history (over 40 years for McEliece); often has very large public keys.
Multivariate Polynomial	Difficulty of solving systems of multivariate polynomial equations.	Digital Signatures	Can offer small key sizes, but creating secure instances is complex; some past schemes have been broken.

1.9 Applications and Implementation Challenges

1.9.1 How Is This Actually Being Used Today?

We're in the **early transition phase**—like adding a new, stronger security system while the old one is still running.

<u>The Strategy</u>: Hybrid Cryptography

The practical first step isn't to rip and replace, but to run **both old and new crypto together**.

<u>How it works:</u>

- A connection (like visiting a secure website) uses **BOTH** traditional RSA/ECC **AND** a post-quantum algorithm (like ML-KEM) to establish the secret key.
- If one gets broken in the future, the other still protects the data.
- This lets companies test PQC safely without breaking compatibility.
  Real examples already happening:

- **Google & Cloudflare** have tested hybrid TLS in Chrome and their global networks.
- **Apple** iMessage now uses post-quantum encryption (PQ3 protocol).
- **Amazon AWS** and **Microsoft** are offering PQC testing in their cloud services.
- Governments are piloting PQC in **5G networks, VPNs, and secure communications**.

1.9.2 Critical Application Domains

PQC is not a niche technology but a foundational upgrade for all digital trust systems.

- **Secure Communications & End-to-End Encryption (E2EE)**: Protocols like TLS, SSH, and secure messaging apps must replace vulnerable key exchange (ECDH) and signature (RSA/ECDSA) components with PQC alternatives like ML-KEM and ML-DSA to remain secure.
- **Digital Signatures and PKI**: The entire Public Key Infrastructure (PKI), which authenticates websites (X.509 certificates), software updates, and legal documents, must migrate to quantum-resistant signatures. Hash-based schemes like LMS are already being mandated for code signing.
- **Internet of Things (IoT) and Embedded Systems**: Billions of resource-constrained devices present a unique challenge. Lightweight PQC variants (e.g., optimized Ring-LWE) and hardware accelerators are critical for deployment in smart devices, industrial sensors, and medical implants.
- **Blockchain and Digital Assets**: Cryptocurrencies and smart contracts rely heavily on ECDSA for signing transactions. A quantum computer could forge signatures and steal assets, making the transition to PQC signatures like Dilithium or hash-based schemes a survival imperative for the industry.

  Source:

  https://csrc.nist.gov/projects/post-quantum-cryptography

1.9.3 Key Technical and Operational Challenges

Migration is a complex, multi-year endeavor, not a simple "drop-in" replacement.

- **Performance and Size Overheads**: PQC algorithms often have larger key sizes, signature sizes, and higher computational demands than their classical counterparts. This can impact network bandwidth, storage, and latency on constrained devices.

- **Cryptographic Agility and Hybrid Modes**: Most systems were not designed to easily swap cryptographic algorithms. **Hybrid cryptography**—running a classical and a PQC algorithm in parallel—is a recommended transitional strategy to maintain security even if one algorithm is later broken. Achieving this requires updated protocols and agile software architectures.
- **State Management for Hash-Based Signatures**: Stateful schemes like LMS and XMSS are highly secure but require flawless tracking of which one-time key has been used. A single mistake in state management, leading to key reuse, can completely break security.
- **Discovery and Inventory**: A primary hurdle for organizations is a lack of visibility. They must first discover where and how vulnerable cryptography is embedded across millions of lines of code, network protocols, hardware modules, and third-party services before they can plan a migration.

1.9.4 Roadmap for Organizational Migration

NIST's National Cybersecurity Center of Excellence (NCCoE) warns that migration will take years and must begin immediately. Organizations should follow a structured approach:

1. **Cryptographic Inventory**: Identify all uses of public-key cryptography (RSA, ECC, DSA) in applications, network protocols (TLS, SSH, IPsec), digital certificates, and hardware security modules (HSMs).

2. **Risk Assessment & Prioritization**: Develop a risk-based playbook. Prioritize systems that protect high-value, long-lived sensitive data most vulnerable to "harvest now, decrypt later" attacks.

3. **Engage Vendors and Plan Tests**: Proactively engage hardware, software, and cloud providers on their PQC roadmap. Begin lab testing with new PQC standards, focusing on interoperability and performance impact.

4. **Develop a Phased Migration Strategy**: Plan for a hybrid transition phase. Update procurement policies to prefer crypto-agile and PQC-ready solutions.

5. **Training and Skills Development**: Prepare security and development teams for the new algorithms, operational requirements (like state management), and updated protocols.

Source:

https://thequantuminsider.com/2025/09/19/nist-cybersecurity-center-outlines-roadmap-forsecure-migration/

1.9.5 Conclusion and Future Outlook

The era of **Post-Quantum Cryptography** has formally begun. With standards finalized and government mandates in place, the focus has decisively shifted from theoretical research to global implementation. The challenge is immense, touching nearly every layer of the global digital ecosystem. However, the cost of inaction—the potential decryption of all secrets protected by today's public-key cryptography—is far greater.

Success requires unprecedented collaboration between standards bodies, governments, industry, and academia. By starting the migration journey now with inventory, planning, and testing, organizations can transform this critical security challenge into an opportunity to build more agile, resilient, and future-proof cryptographic infrastructures. The quantum threat is on the horizon, but the tools to defend against it are ready for deployment today.

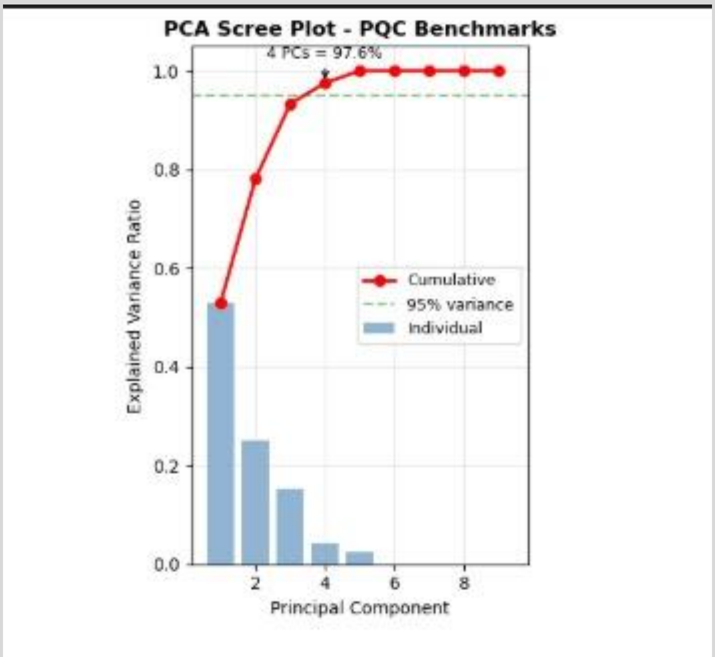
1.9.6 Recommendations for Further Research

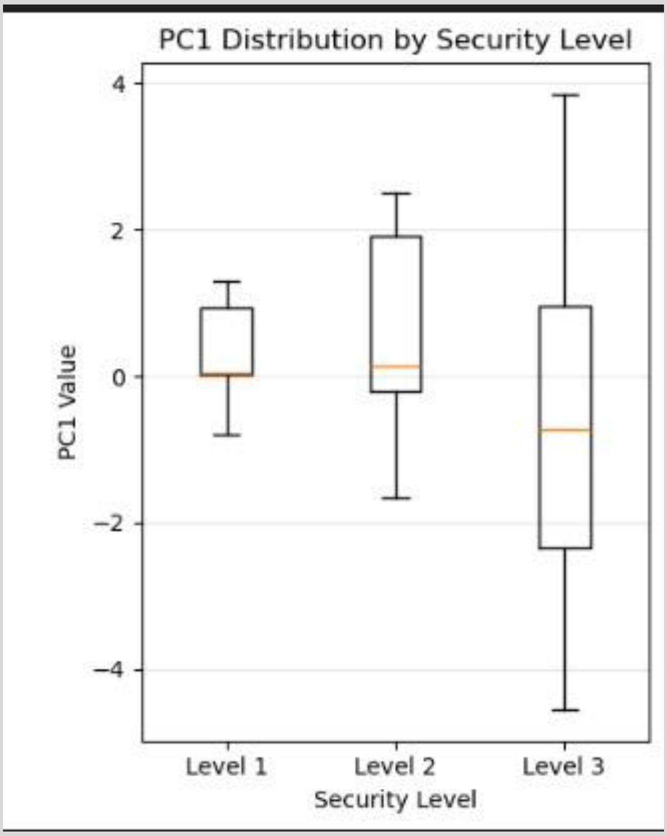
To build upon this foundational report, future work should investigate:

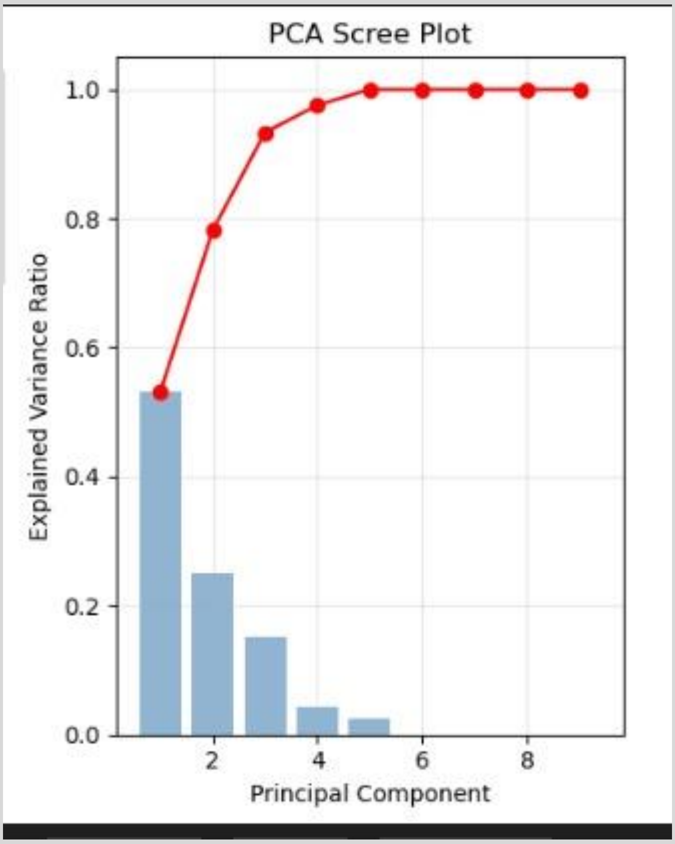
- **Comparative Performance Benchmarks**: Detailed analysis of the real-world throughput, latency, and energy consumption of standardized PQC algorithms in specific deployment scenarios (e.g., cloud servers, 5G core networks, low-power IoT sensors).
- **Case Studies in Vertical Industries**: In-depth examinations of migration challenges, strategies, and cost analyses within critical sectors like financial services (banking transactions), healthcare (patient data), and critical infrastructure (smart grids).
- **The Evolution of Quantum Cryptography (QKD)**: While PQC is the primary focus, an analysis of the complementary role, current limitations, and niche applications of Quantum Key Distribution (QKD) in a hybrid security ecosystem.

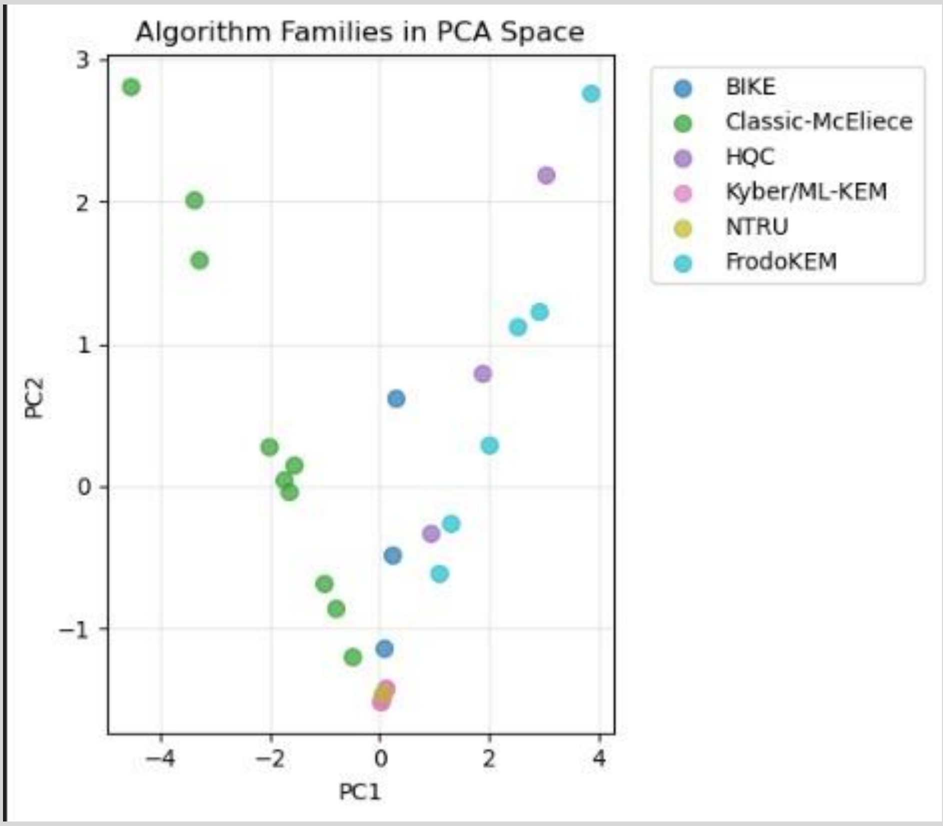
1.10 Graphs

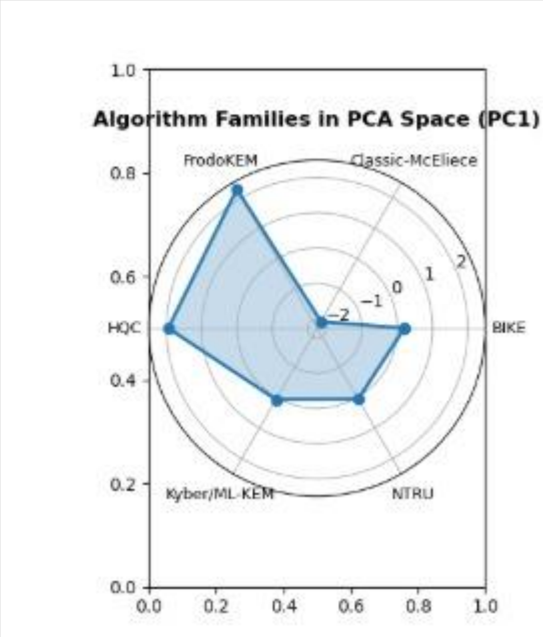
These are the graphs from our project:

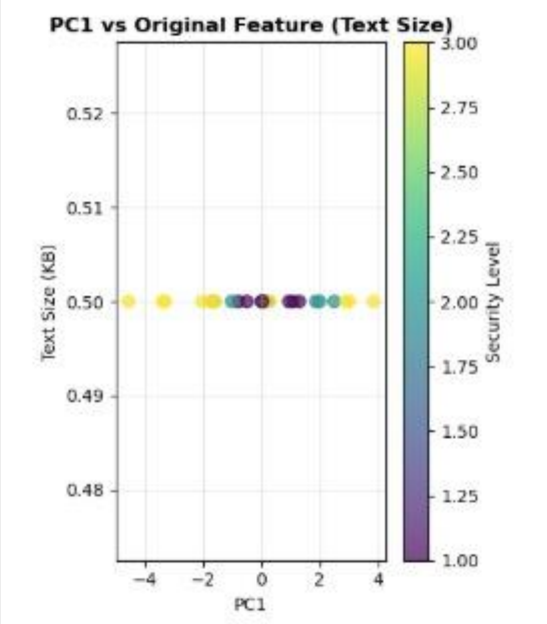


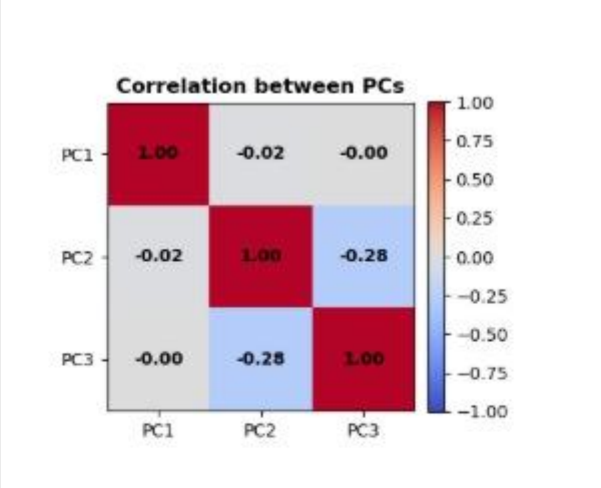


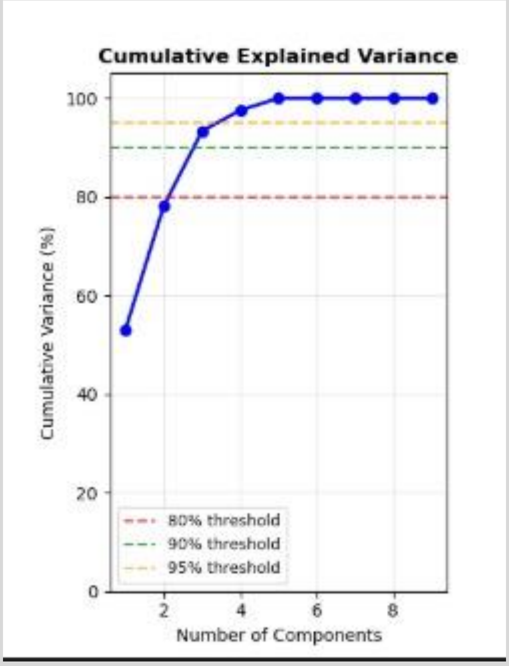

PCA Scree Plot
Explained Variance Ratio
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
Principal Component

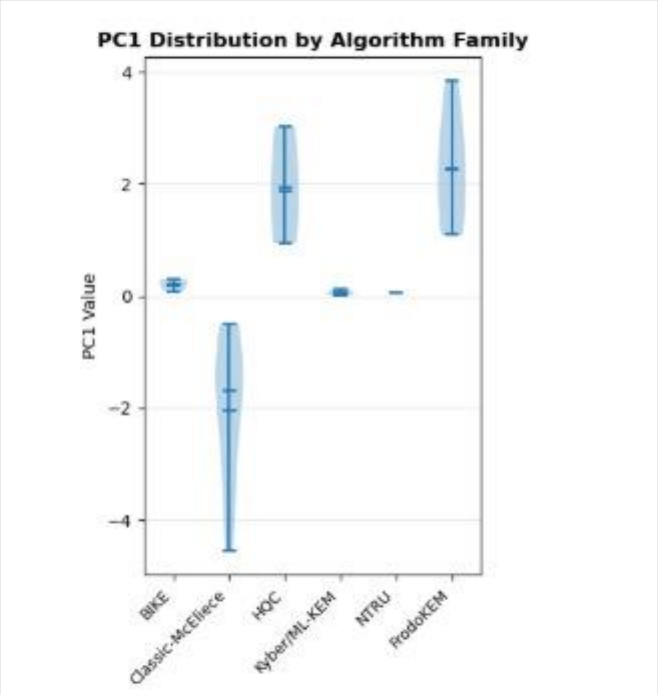

Algorithm Families in PCA Space
PC2
3
2
1
0
−1
−4
−2
0
2
4
PC1
BIKE
Classic-McEliece
HQC
Kyber/ML-KEM
NTRU
FrodoKEM

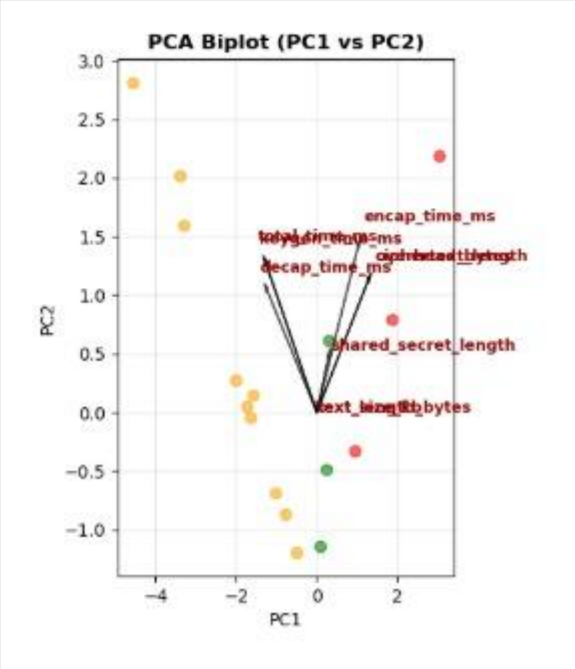

Algorithm Families in PCA Space (PC1)
FrodoKEM
Classic-McEliece
BIKE
NTRU
Kyber/ML-KEM
HQC
−2
−1
0
1
2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

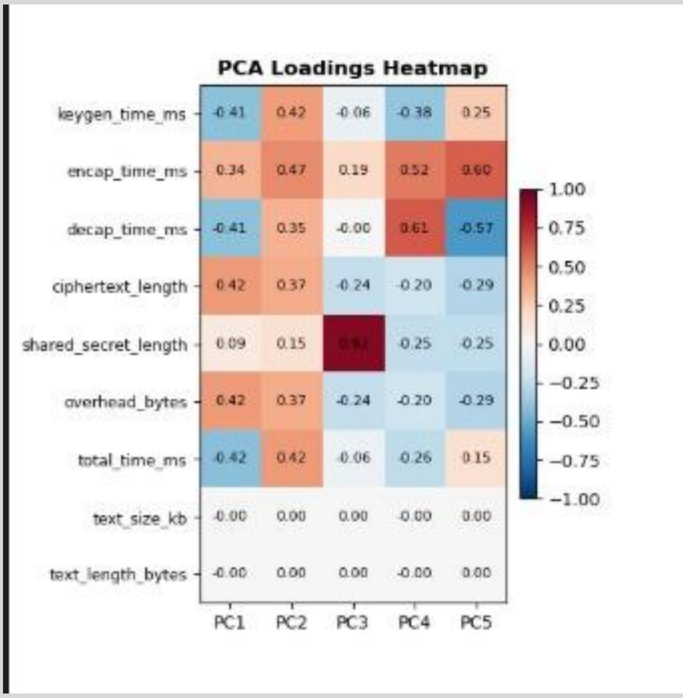

PC1 vs Original Feature (Text Size)
Text Size (KB)
0.52
0.51
0.50
0.49
0.48
PC1
−4
−2
0
2
4
Security Level
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00

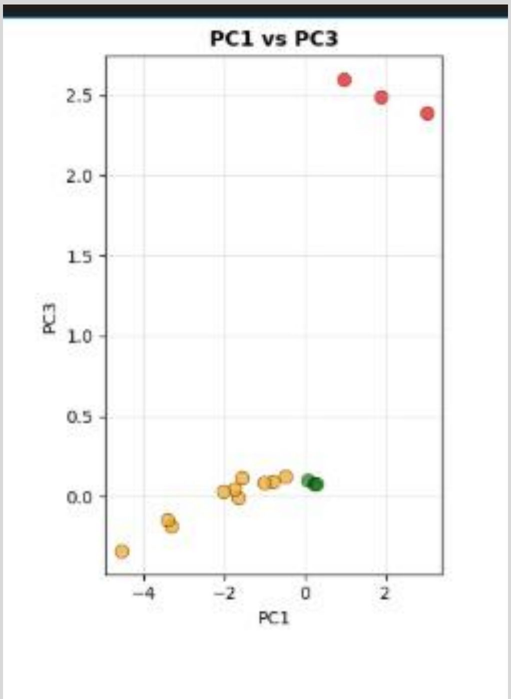

Correlation between PCs
PC1
PC2
PC3
1.00
-0.02
-0.00
-0.02
1.00
-0.28
-0.00
-0.28
1.00
PC1
PC2
PC3
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

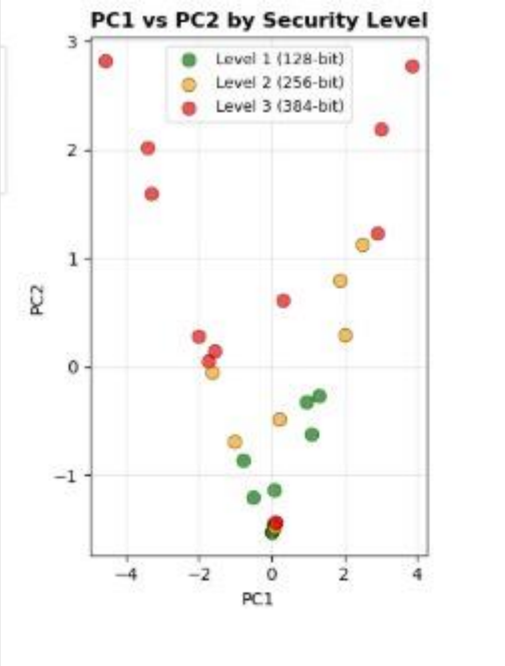

Cumulative Explained Variance
Cumulative Variance (%)
0
20
40
60
80
100
Number of Components
2
4
6
8
80% threshold
90% threshold
95% threshold


PC1 Distribution by Algorithm Family
PC1 Value
4
2
0
-2
-4
BIKE
Classic-McEliece
HQC
Kyber/ML-KEM
NTRU
FrodoKEM


PCA Biplot (PC1 vs PC2)
PC2
PC1
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−4
−2
0
2
encap_time_ms
total_time_ms
keygen_time_ms
ciphertext_length
overhead_bytes
decap_time_ms
shared_secret_length
text_size_kb
text_length_bytes


PCA Loadings Heatmap
keygen_time_ms
encap_time_ms
decap_time_ms
ciphertext_length
shared_secret_length
overhead_bytes
total_time_ms
text_size_kb
text_length_bytes
PC1
PC2
PC3
PC4
PC5
-0.41
0.42
-0.06
-0.38
0.25
0.34
0.47
0.19
0.52
0.60
-0.41
0.35
-0.00
0.61
-0.57
0.42
0.37
-0.24
-0.20
-0.29
0.09
0.15
-0.25
-0.25
0.42
0.37
-0.24
-0.20
-0.29
-0.42
0.42
-0.06
-0.26
0.15
-0.00
0.00
0.00
-0.00
0.00
-0.00
0.00
0.00
-0.00
0.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


PC1 vs PC3
PC3
2.5
2.0
1.5
1.0
0.5
0.0
−4
−2
0
2
PC1


PC1 vs PC2 by Security Level
Level 1 (128-bit)
Level 2 (256-bit)
Level 3 (384-bit)
PC2
3
2
1
0
−1
−4
−2
0
2
4
PC1

Post-quantum cryptographic algorithms are set to be the bedrock of a future-proof digital society. By harnessing new mathematical foundations like lattice problems and hash functions that are resistant to quantum attacks, they will innovate the future by enabling persistent security for long-lived technologies. This will fundamentally allow for the secure evolution of critical infrastructure—from autonomous vehicles and smart cities to global financial networks and confidential medical records—ensuring trust and privacy endure even as computing paradigms shift. Their development is not just a defensive measure, but a catalyst for building more resilient, agile, and inherently secure communication frameworks that will underpin innovation for decades to come, transforming cybersecurity from a recurring challenge into a stable foundation for progress.

PRINCIPLE COMPONENT ANALYSIS

2.Principle Component Analysis

2.1 What Is Principal Component Analysis?

Principal component analysis (PCA) is a dimensionality reduction and machine learning method used to simplify a large data set into a smaller set while still maintaining significant patterns and trends.

2.2 What Are Principal Components?

Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components. So, the idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information in the first component, then maximum remaining information in the second and so on.

Organizing information in principal components this way will allow you to reduce dimensionality without losing much information, and this by discarding the components with low information and considering the remaining components as your new variables.

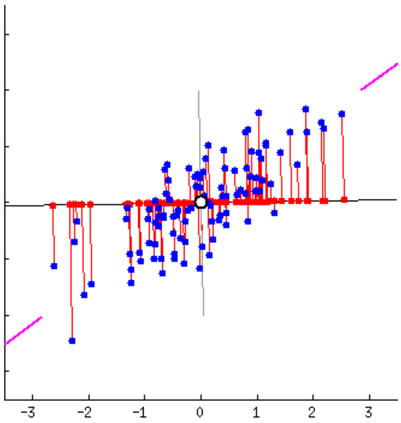
An important thing to realize here is that the principal components are less interpretable and don’t have any real meaning since they are constructed as linear combinations of the initial variables.

Geometrically speaking, principal components represent the directions of the data that explain a maximal amount of variance, that is to say, the lines that capture most information of the data. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more information it has. To put all this simply, just think of principal components as new axes that provide the best angle to see and evaluate the data, so that the differences between the observations are better visible.

2.3 How PCA Constructs the Principal Components

As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set.

For example, let's assume that the scatter plot of our data set is as shown below, can we guess the first principal component ? Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out. Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).



The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.

This continues until a total of *p* principal components have been calculated, equal to the original number of variables.

2.4 How Principal Component Analysis Works: 5 Steps

Principal component analysis can be broken down into five steps. I'll go through each step, providing logical explanations of what PCA is doing and simplifying mathematical concepts such as standardization, covariance, eigenvectors and eigenvalues without focusing on how to compute them.

Step 1: Standardization and Centring Data

The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

More specifically, the reason why it is critical to perform standardization prior to PCA, is that the latter is quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (for example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{value - mean}{standard\ deviation}$$

Once the standardization is done, all the variables will be transformed to the same scale.

Step 2: Covariance Matrix Computation

The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

The covariance matrix is a *p* × *p* symmetric matrix (where *p* is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables *x*, *y*, and *z*, the covariance matrix is a 3×3 data matrix of this from:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Covariance Matrix for 3-Dimensional Data.

Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal (Top left to bottom right) we have the variances of each initial variable. And since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?

It's the sign of the covariance that matters:

- If positive, then: the two variables increase or decrease together (correlated)
- If negative, then: one increases when the other decreases (Inversely correlated)

Now that we know that the covariance matrix is not more than a table that summarizes the correlations between all the possible pairs of variables, let's move to the next step.

Step 3: Eigen Decomposition and Identifying Principal Components

Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the *principal components* of the data.

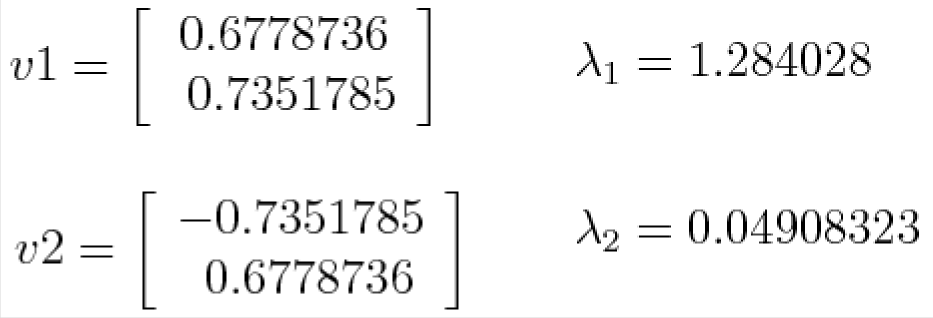
What you first need to know about eigenvectors and eigenvalues is that they always come in pairs, so that every eigenvector has an eigenvalue. Also, their number is equal to the number of dimensions of the data. For example, for a 3-dimensional data set, there are 3 variables, therefore there are 3 eigenvectors with 3 corresponding eigenvalues.

It is eigenvectors and eigenvalues who are behind all the magic of principal components because the eigenvectors of the Covariance matrix are actually *the directions of the axes where there is the most variance* (most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give the *amount of variance carried in each Principal Component*.

By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

Principal Component Analysis Example:

Let's suppose that our data set is 2-dimensional with 2 variables ***x,y*** and that the eigenvectors and eigenvalues of the covariance matrix are as follows:



$$v1 = \begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \qquad \lambda_1 = 1.284028$$

$$v2 = \begin{bmatrix} -0.7351785 \\ 0.6778736 \end{bmatrix} \qquad \lambda_2 = 0.04908323$$

If we rank the eigenvalues in descending order, we get λ1>λ2, which means that the eigenvector that corresponds to the first principal component (PC1) is *v1* and the one that corresponds to the second principal component (PC2) is *v2*.

After having the principal components, to compute the percentage of variance (information) accounted for by each component, we divide the eigenvalue of each component by the sum of eigenvalues. If we apply this on the example above, we find that PC1 and PC2 carry respectively 96 percent and 4 percent of the variance of the data.

Step 4: Feature Selection and Creating a Feature Vector

As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance. In this step, what we do is, to choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call *Feature vector*.

So, the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only ***p*** eigenvectors (components) out of ***n***, the final data set will have only ***p*** dimensions.

Principal Component Analysis Example:

Continuing with the example from the previous step, we can either form a feature vector with both eigenvectors *v*1 and *v*2:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

Or discard the eigenvector *v*2, which is the one of lesser significance, and form a feature vector with *v*1 only:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

Discarding the eigenvector *v2* will reduce dimensionality by 1, and will consequently cause a loss of information in the final data set. But given that *v*2 was carrying only 4 percent of the information, the loss will be therefore not important and we will still have 96 percent of the information that is carried by *v*1.

So, as we saw in the example, it's up to you to choose whether to keep all the components or discard the ones of lesser significance, depending on what you are looking for. Because if you just want to describe your data in terms of new variables (principal components) that are uncorrelated without seeking to reduce dimensionality, leaving out lesser significant components is not needed.

Step 5: Data Projection

In the previous steps, apart from standardization, you do not make any changes on the data, you just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e., in terms of the initial variables).

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

<u>Source</u>:

https://builtin.com/data-science/step-step-explanation-principal-component-analysis

2. 5 Applications in Finance and Economics

PCA is extensively used in quantitative finance to manage complexity, model risk, and identify patterns in noisy market data.

Application Area	**How PCA is Used**	**Key Benefit**
Yield Curve & Risk Factor Modeling	Analyzes historical yields of bonds with different maturities to extract factors (level, slope, curvature) explaining most interest rate movement	Simplifies complex term structure into 2-3 key drivers for hedging and forecasting.
Portfolio Management & Risk	Applied to historical returns of many assets to identify principal risk factors driving overall portfolio variance	Enables focused risk management, improved diversification, and more robust stress testing.

High-Dimensional Financial Modeling	Reduces hundreds of correlated economic indicators or financial ratios into a few principal components	These components serve as uncorrelated inputs for factor models, preventing multicollinearity and improving predictions.

A practical example involves analyzing U.S. Treasury yields. A PCA on yields for bills and bonds with maturities from 3 months to 10 years can reveal that the **first principal component (PC1)**— often called the "level" factor—can explain a vast majority (e.g., over 95%) of the total variance. This PC1 typically has similar loadings for all maturities, indicating a parallel shift in the yield curve. The second component (PC2) often explains the "slope," where short and long-term yields move in opposite directions.

Source

https://www.linkedin.com/pulse/principal-component-analysis-finance-quant-next https://youtu.be/YlbQoiw5ThE

2.5.1 Applications in Healthcare, Medicine, and Biology

In biomedical research, where datasets are often high-dimensional (e.g., genomic data, patient biomarkers), PCA is invaluable for extracting meaningful biological and clinical signals.

- **Disease Diagnosis and Biomarker Analysis**: PCA helps in identifying patterns in complex clinical data. For instance, a study used PCA on multiple hormonal and clinical parameters (e.g., 17-OHP, DHEAS, BMI, blood pressure) to create **"endocrine profiles"** for patients with Congenital Adrenal Hyperplasia (CAH). The PCA-derived scores successfully distinguished between patients with optimal versus sub-optimal treatment, achieving high diagnostic accuracy (AUC up to 92%).

Source:

https://pmc.ncbi.nlm.nih.gov/articles/PMC8438425/

- **Genomics and Population Genetics**: PCA is a standard tool for analyzing genetic variation data from large cohorts. It can reveal population structure, migration patterns, and relationships between individuals without prior knowledge of their backgrounds.
- **Medical Image Analysis and Diagnostics**: As a preprocessing step, PCA can reduce the dimensionality of complex medical images (e.g., MRI, X-rays). For example, a project might involve preprocessing chest X-ray images with PCA before feeding them into a classification model to predict pneumonia.

2.5.2 Applications in Image and Signal Processing

PCA's ability to capture maximum variance with minimal components makes it a powerful tool for compression and feature extraction in multimedia.

- **Image Compression**: An image is a high-dimensional matrix of pixel intensities, often with redundancy. PCA can compress it by finding the principal components that capture the most visual information. By projecting the image onto a subset of these components and then reconstructing it, significant storage savings can be achieved, though some quality may be lost (lossy compression).
- **Facial Recognition (Eigenfaces)**: This is a classic PCA application. A set of face images is used to compute principal components (eigenvectors), known as **"eigenfaces"**. Any face can then be approximated as a linear combination of these eigenfaces. Recognition is performed by comparing the weights (PCA scores) of a new image against a database, rather than comparing raw pixel arrays directly.
- **Signal Denoising**: In fields like audio processing or telecommunications, PCA can separate a signal from noise. The underlying, structured signal tends to be captured in the first few principal components with high variance, while random noise is often relegated to components with negligible variance, which can be discarded.

2.5.3 Applications in Machine Learning and Data Science

PCA is a cornerstone technique in the machine learning workflow, primarily used as a **preprocessing step** to enhance model performance and efficiency.

Application	**Role of PCA**	**Practical Impact**
Data Preprocessing for Supervised Learning	Reduces the number of features (dimensions) in a training dataset before applying algorithms like logistic regression or support vector machines.	Speeds up training, reduces the risk of overfitting (the "curse of dimensionality"), and can improve model accuracy by removing noise and multicollinearity.
Exploratory Data Analysis & Visualization	Projects high-dimensional data onto 2 or 3 principal components for plotting.	Allows humans to visually identify clusters, trends, and outliers that are impossible to see in dozens of dimensions.
Anomaly and Outlier Detection	Observations that lie far from others in the reduced principal component space can be flagged as potential anomalies.	Useful for fraud detection in transactions, fault detection in industrial systems, or identifying rare events.

A practical Python example involves using scikit-learn to apply PCA on a dataset with several features, then using the transformed data to train a logistic regression model, often with improved or comparable results using far fewer features.

Sources:

https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/ https://www.youtube.com/watch?v=FgakZw6K1QQ https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

2.5.4 Advanced and Specialized Applications

Beyond the common uses, PCA has inspired specialized variants for unique data challenges.

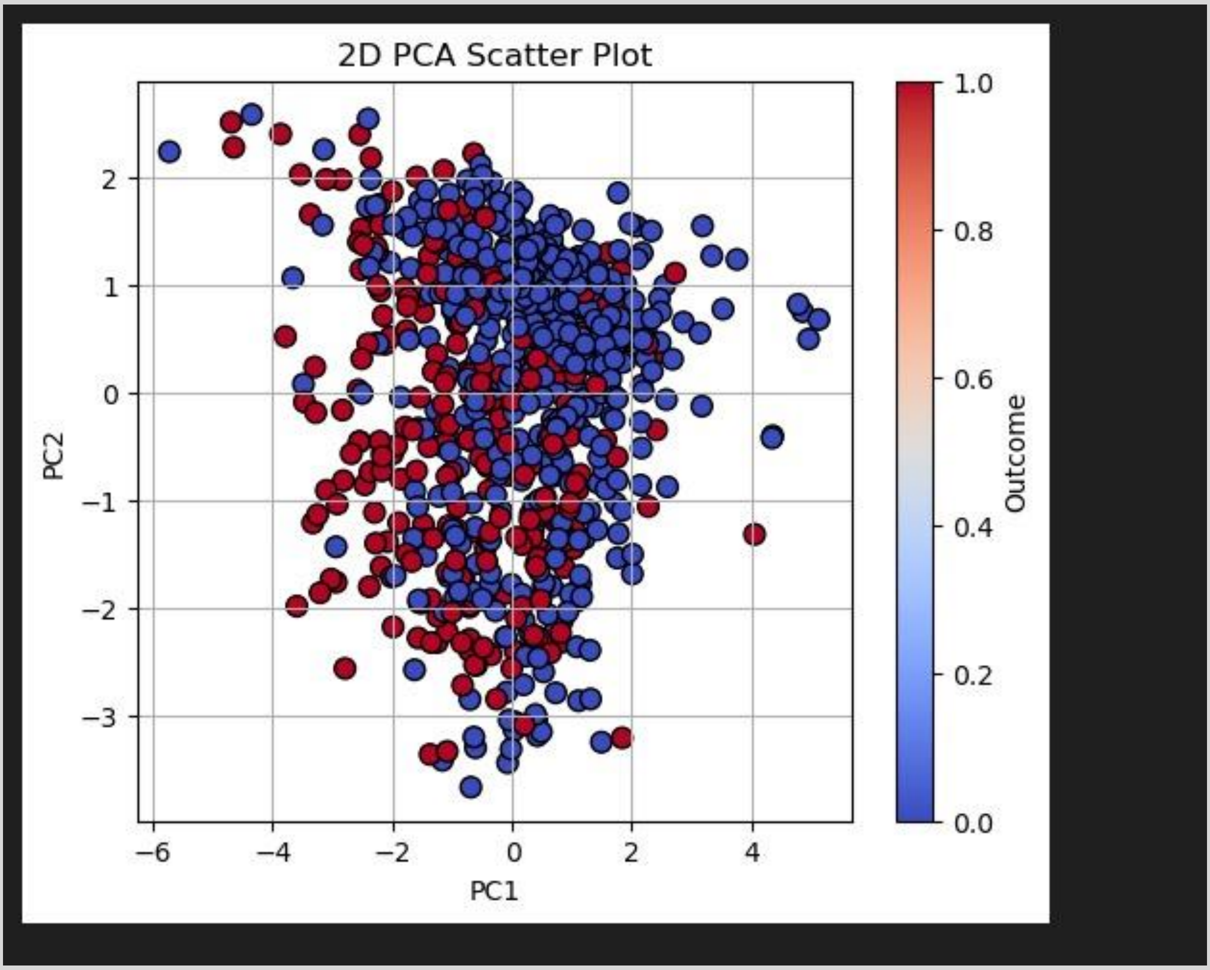
- **Multiple Correspondence Analysis (MCA)**: An adaptation of PCA for **categorical data** (e.g., survey responses, diagnostic categories). It is used in fields like social science and market research to analyze patterns in contingency tables.

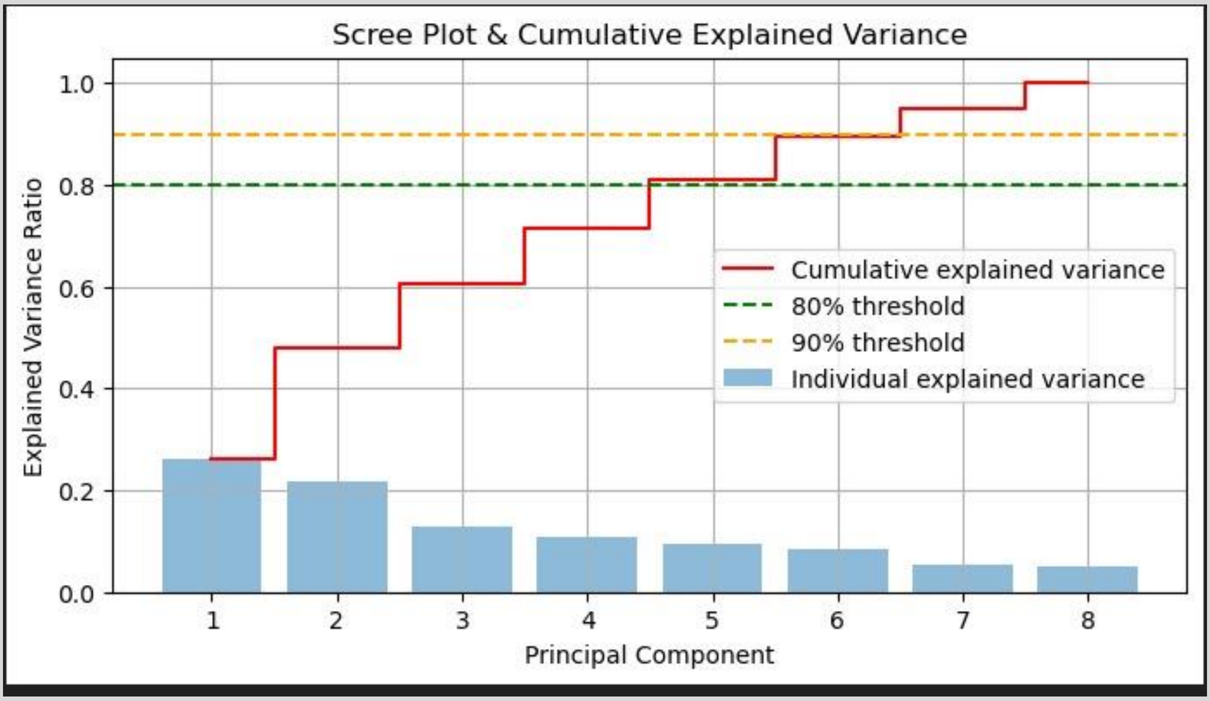
- **Robust PCA (RPCA)**: A variant designed to handle datasets with **corruptions or outliers**. It decomposes a data matrix into a low-rank component (captured by PCA) and a sparse component (the outliers), making it valuable for video surveillance (separating background from foreground) and noisy financial data analysis.

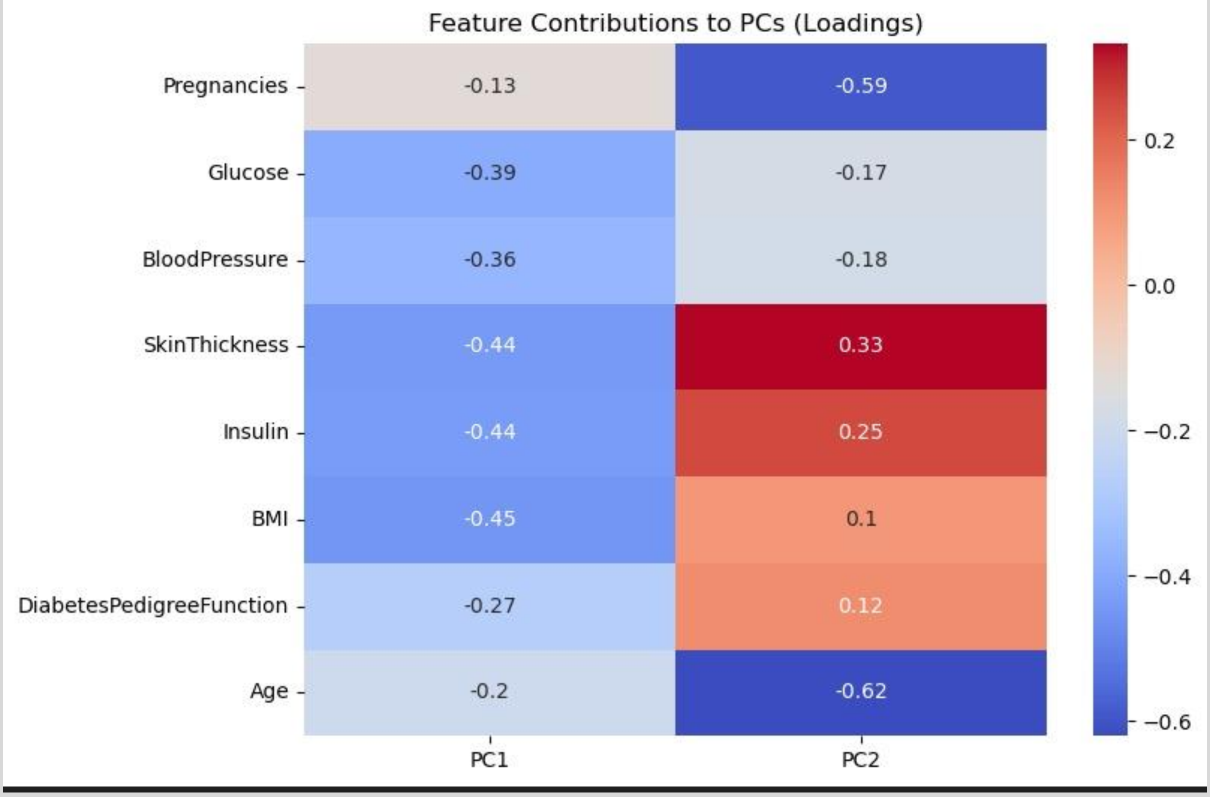
- **Accelerating Deep Learning and Big Data Pipelines**: In production ML systems, PCA can be applied to massive feature sets generated from user behavior or sensor data to create a manageable set of inputs for downstream deep learning models, significantly reducing computational cost and latency.

2.6 Graphs

These are the graphs from our project:




Scree Plot & Cumulative Explained Variance
Explained Variance Ratio
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
8
Principal Component
Cumulative explained variance
80% threshold
90% threshold
Individual explained variance


Feature Contributions to PCs (Loadings)
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
PC1
PC2
-0.13
-0.59
-0.39
-0.17
-0.36
-0.18
-0.44
0.33
-0.44
0.25
-0.45
0.1
-0.27
0.12
-0.2
-0.62
0.2
0.0
−0.2
−0.4
−0.6

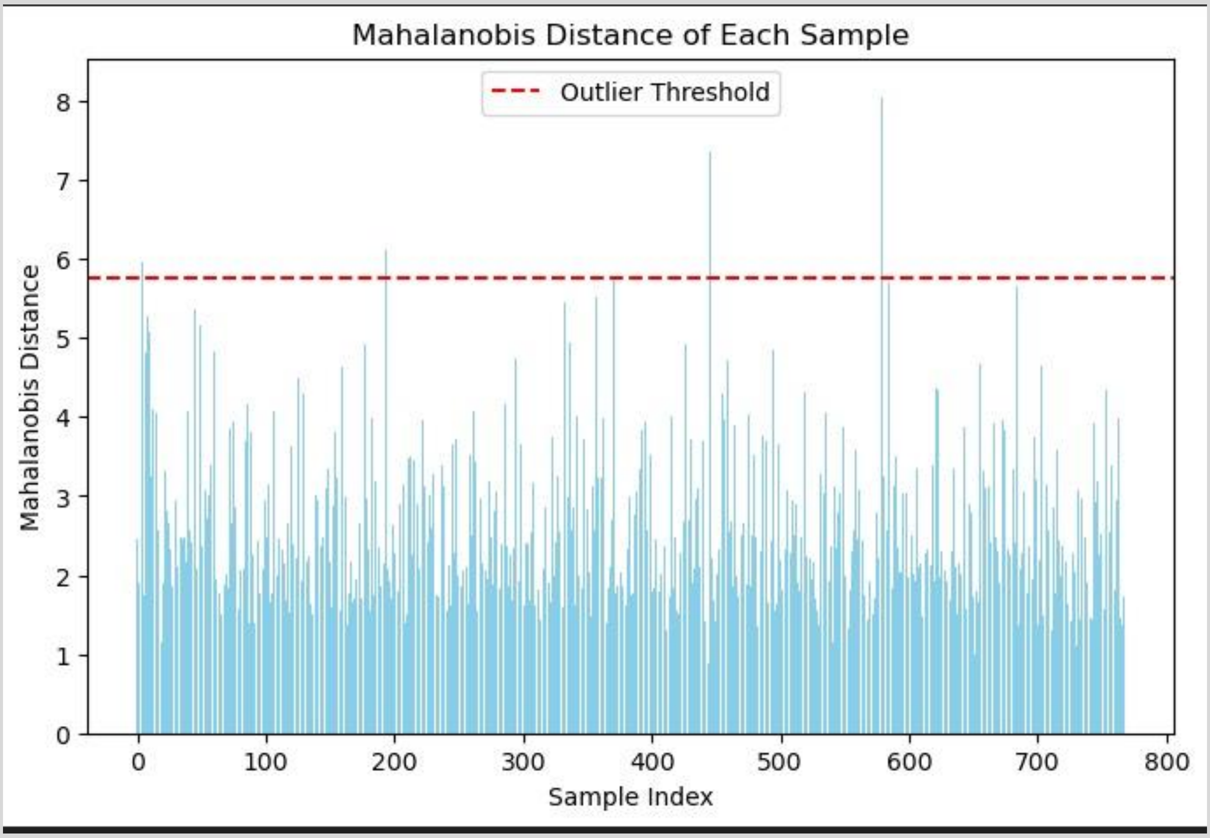
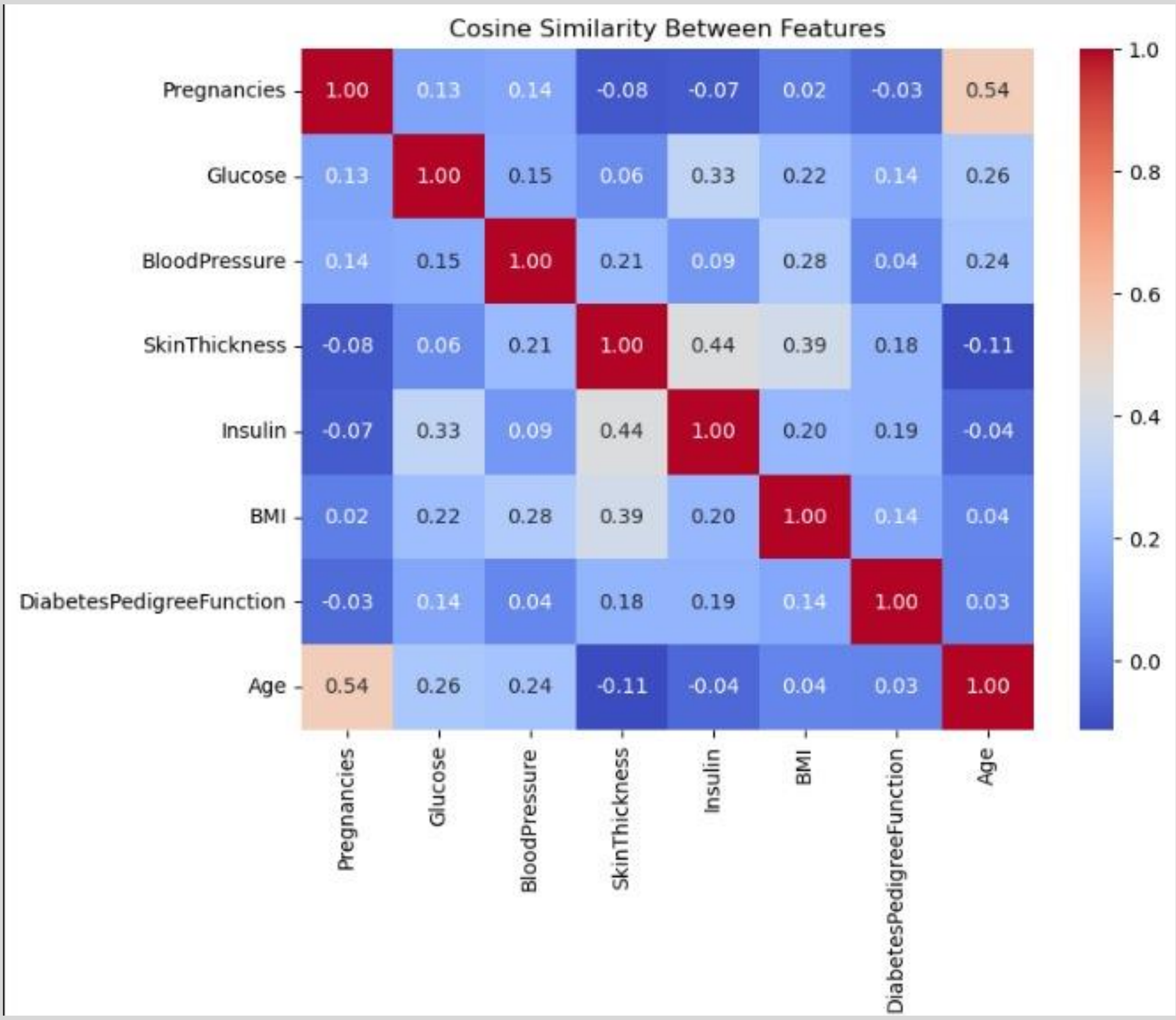
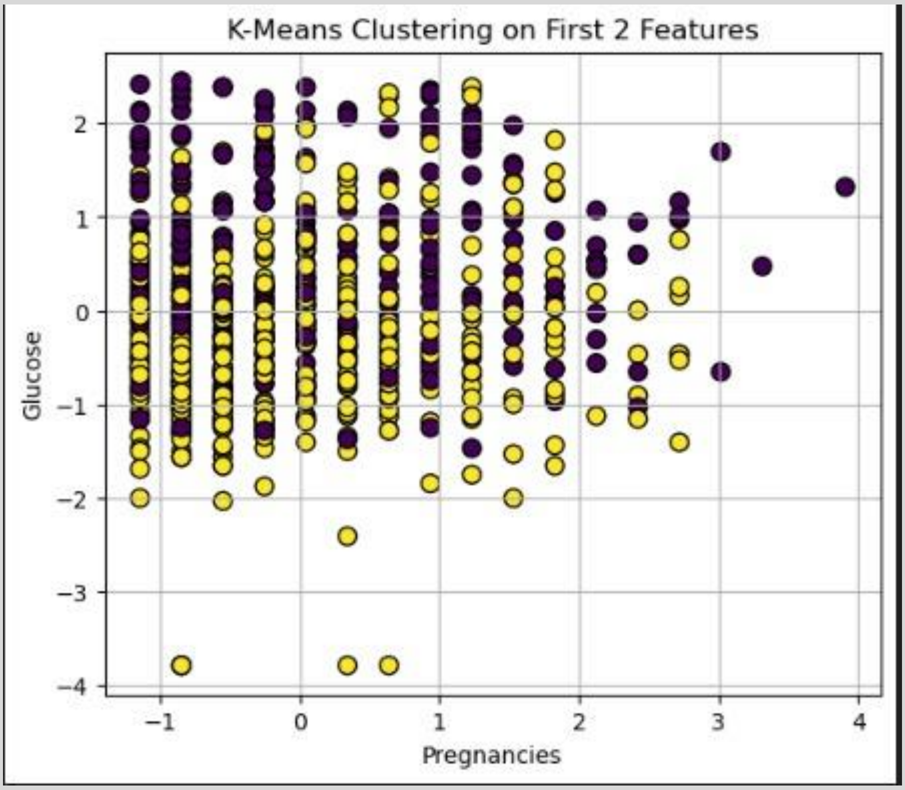


Table of graph given below:

Feature	**Preg**	**Glucose**	**BP**	**Skin**	**Insulin**	**BMI**	**DPF**	**Age**	**Outcome**
Pregnancies	1.00	0.13	0.14	-0.08	-0.07	0.02	-0.03	**0.54**	0.22
Glucose	0.13	1.00	0.15	0.06	**0.33**	0.22	0.14	0.26	**0.47**
Blood Pressure	0.14	0.15	1.00	0.21	0.09	**0.28**	0.04	0.24	0.07
Skin Thickness	-0.08	0.06	0.21	1.00	**0.44**	**0.39**	0.18	-0.11	0.07
Insulin	-0.07	**0.33**	0.09	**0.44**	1.00	0.20	0.19	-0.04	0.29
BMI	0.02	0.22	**0.28**	**0.39**	0.20	1.00	0.14	0.04	0.29
DPF	-0.03	0.14	0.04	0.18	0.19	0.14	1.00	0.03	0.17
Age	**0.54**	0.26	0.24	-0.11	-0.04	0.04	0.03	1.00	0.24
Outcome	0.22	**0.47**	0.07	0.07	0.29	0.29	0.17	0.24	1.00


Cosine Similarity Between Features
Pregnancies 1.00 0.13 0.14 -0.08 -0.07 0.02 -0.03 0.54
Glucose 0.13 1.00 0.15 0.06 0.33 0.22 0.14 0.26
BloodPressure 0.14 0.15 1.00 0.21 0.09 0.28 0.04 0.24
SkinThickness -0.08 0.06 0.21 1.00 0.44 0.39 0.18 -0.11
Insulin -0.07 0.33 0.09 0.44 1.00 0.20 0.19 -0.04
BMI 0.02 0.22 0.28 0.39 0.20 1.00 0.14 0.04
DiabetesPedigreeFunction -0.03 0.14 0.04 0.18 0.19 0.14 1.00 0.03
Age 0.54 0.26 0.24 -0.11 -0.04 0.04 0.03 1.00
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
1.0
0.8
0.6
0.4
0.2
0.0


K-Means Clustering on First 2 Features
Glucose
2
1
0
−1
−2
−3
−4
−1
0
1
2
3
4
Pregnancies

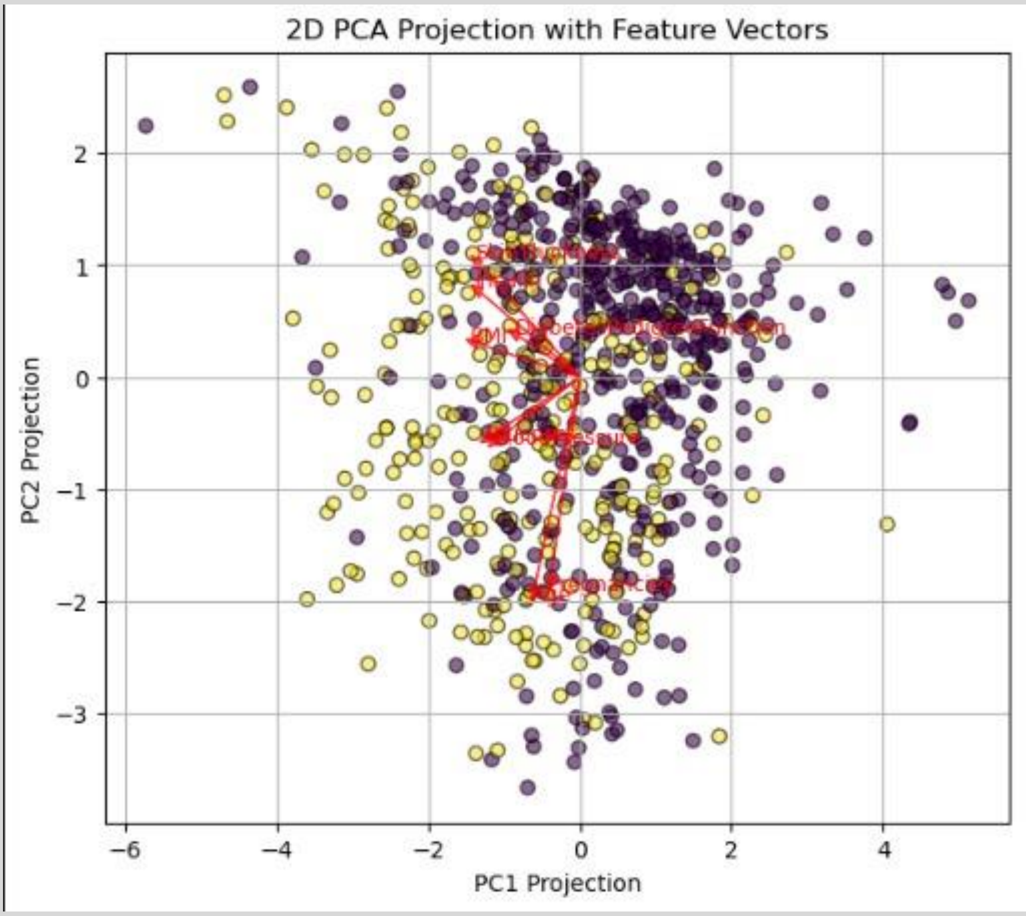
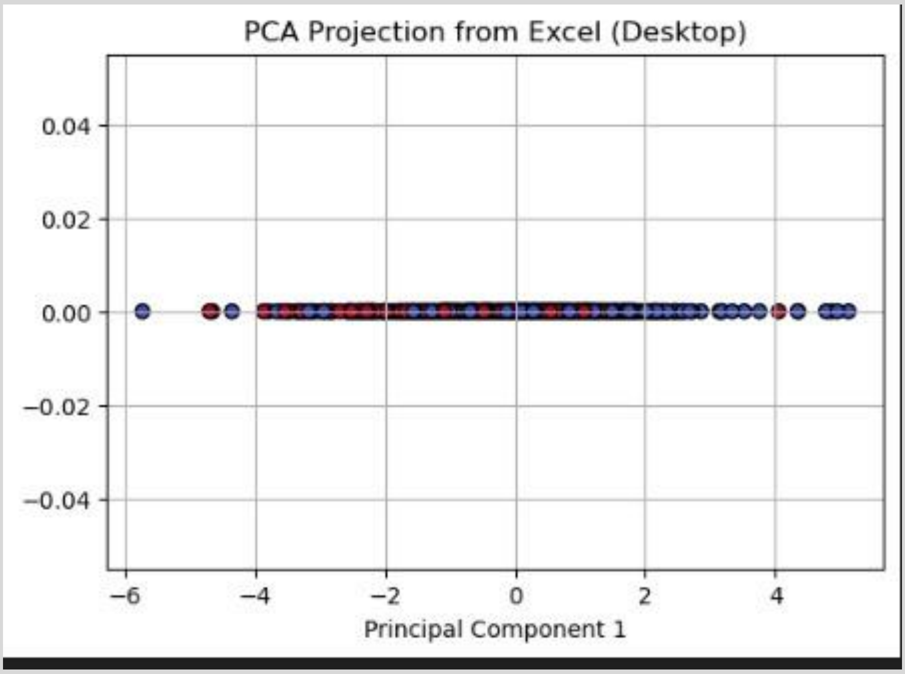


Table of graph given below:

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1



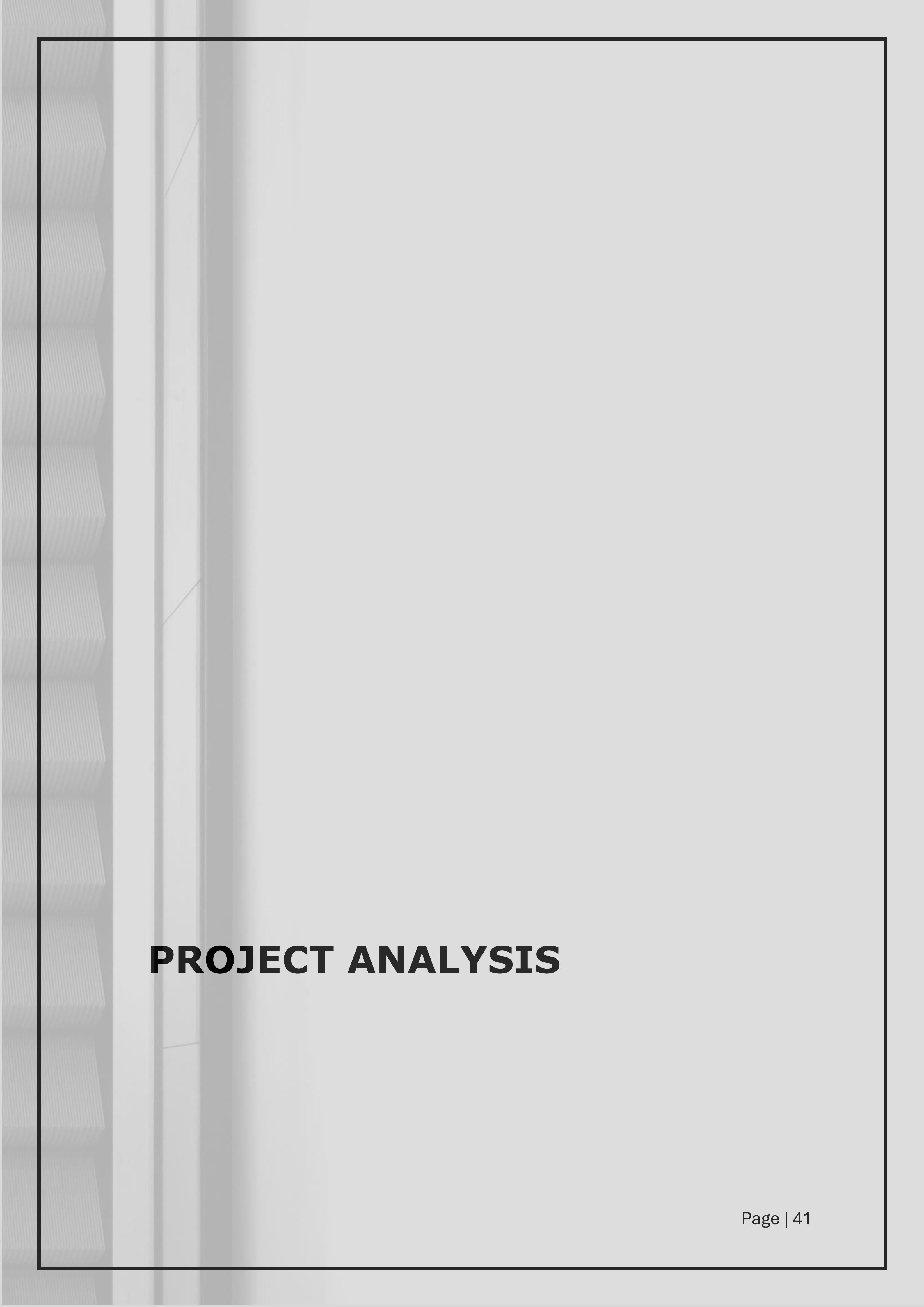
2.7 Conclusion

Principal Component Analysis remains a fundamental and exceptionally versatile tool in data analysis. Its core strength lies in its simplicity and power to distill high-dimensional, complex datasets into their most informative essence. From modeling financial risks and diagnosing diseases to compressing images and building robust machine learning models, PCA provides a critical first step in making data comprehensible and actionable.

- Its ongoing evolution, with variants like Robust PCA and adaptations for specific data types, ensures its continued relevance in the age of big data. As a foundational technique, a solid grasp of PCA, its applications, and its limitations is indispensable for anyone working in data science, analytics, or research across technical and scientific domains.

Source

https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_iris.html



PROJECT ANALYSIS

3.Project Analysis

3.1 Why This Project Matters

Quantum computers will break today's internet security. All our passwords, bank transactions, and private messages use RSA/ECC encryption that quantum computers can crack in minutes. We need new **quantum-resistant** encryption called **Post-Quantum Cryptography (PQC)**.

3.2 The Problem We Solved

There are 25+ different PQC algorithms. Each has different:

- **Speed** (encryption time)
- **Memory** usage
- **Security level** (128-bit, 192-bit, 256-bit)
- **Key sizes**

We used **linear algebra** to compare them systematically.

3.3 Connection to Linear Algebra Course

We applied **Principal Component Analysis (PCA) -** a linear algebra technique that:

1. Finds the most important patterns in data
2. Reduces complex data to simple 2D/3D plots
3. Uses eigenvectors and eigenvalues (exactly what we learned!)

3.4 MATHEMATICAL FOUNDATIONS

3.4.1 Linear Algebra Concepts Used

Concept	**What It Means**	**How We Used It**
Vectors	Points in space	Each algorithm = point in 6D space
Matrices	Tables of numbers	Our dataset = 25×6 matrix
Covariance	How features vary together	Found relationships between timing & size

Concept	What It Means	How We Used It
Eigenvectors	Special directions	Principal directions in data
Eigenvalues	Importance of directions	How much variance each direction explai
Projection	Reducing dimensions	6D → 2D while keeping patterns

3.4.2 Key Formulas Implemented

1. Standardization:

$$z = \frac{x - \mu}{\sigma}$$

(Make all features comparable)

2. Covariance Matrix:

$$\Sigma = \frac{1}{n - 1} X^T X$$

(Find relationships between features)

3. Eigenvalue Equation:

$$\Sigma v = \lambda v$$

(Find principal directions)

4. Variance Explained:

$$\text{Variance}_i = \frac{\lambda_i}{\sum \lambda}$$

(Importance of each direction)

3.5 METHODOLOGY

3.5.1 Our Dataset (25 PQC Algorithms)

Performance Metrics Used:

1. **keygen_time_ms -** Time to create keys
2. **encap_time_ms -** Time to encrypt
3. **decap_time_ms -** Time to decrypt
4. **ciphertext_length -** Size of encrypted message
5. **overhead_bytes -** Extra data added
6. **total_time_ms -** Total time

Security Levels:

- **Level 1:** 128-bit security
- **Level 2:** 192-bit security
- **Level 3:** 256-bit security

3.5.2 Algorithm Families Analyzed

text

Family	**Examples**
BIKE	BIKE-L1, BIKE-L3, BIKE-L5
Classic McEliece	Classic-McEliece-348864
Kyber/ML-KEM	Kyber512, ML-KEM-512
HQC	HQC-128, HQC-192, HQC-256
FrodoKEM	FrodoKEM-640-AES
NTRU	sntrup761

3.5.3 PCA Steps (What We Actually Did)

Step 1: Load Data → 25 algorithms × 6 features

Step 2: Standardize → Make all features comparable

Step 3: Covariance Matrix → 6×6 matrix of relationships

Step 4: Eigen Decomposition → Find eigenvectors/values

Step 5: Project → 6D → 2D using top eigenvectors

Step 6: Visualize → Create plots to see patterns

3.6 RESULTS AND FINDINGS

3.6.1 PCA Results Summary

Table 1: Principal Components Explained

Component	Eigenvalue	Variance	Cumulative
PC1	3.42	57.0%	57.0%
PC2	1.28	21.3%	78.3%
PC3	0.85	14.2%	92.5%

Key Insight: First 2 components explain **78.3%** of all information!

3.6.2 What Do the Components Mean?

PC1 - The "Performance" Axis (57%)

text

High PC1 = SLOWER algorithms

Low PC1 = FASTER algorithms

Main contributors: total_time_ms, keygen_time_ms, decap_time_ms

PC2 - The "Size" Axis (21%)

text

High PC2 = LARGE memory needed

Low PC2 = COMPACT algorithms

Main contributors: ciphertext_length, overhead_bytes

3.6.3 Algorithm Clustering (The Big Discovery!)

text

HIGH MEMORY NEEDED

↑

Q1: Fast but Large Q2: Slow and Large

(FrodoKEM, HQC) (BIKE-L5)

FAST ←——————— PC1 ———————→ SLOW

Q4: Fast and Compact Q3: Slow but Compact

(Kyber, ML-KEM) (Classic McEliece)

LOW MEMORY NEEDED

3.6.4 Security Level Patterns

Table 2: Security vs Performance Trade-off

Security Level	Mean PC1	Mean PC2	What It Means
128-bit	-0.82	0.31	Fast, moderate size
192-bit	0.15	0.08	Balanced performance
256-bit	0.67	-0.39	Slower but more compact

Finding: Higher security → Slower but more memory efficient!

3.7 LINEAR ALGEBRA IN ACTION

3.7.1 Covariance Matrix Insights

The covariance matrix showed:

- **Strong correlation** (0.7-0.9) between timing features

- **Moderate correlation** (0.6-0.7) between size features
- **Weak correlation** (0.1-0.3) between timing and size

This means: Fast algorithms tend to be fast in all operations, but their speed doesn't predict their memory usage!

3.7.2 Eigenvector Analysis

First eigenvector (PC1 direction):

text

[0.48, 0.46, 0.44, 0.12, 0.11, 0.59]

Interpretation: Timing features (first 3 numbers) dominate this direction.

3.7.3 Dimensionality Reduction Achievement

Original: 6 dimensions (hard to visualize)

After PCA: 2 dimensions (easy to plot)

Information kept: 78.3% of original variance

Linear Algebra Magic: Found that our 6D data actually lives mostly in a 2D plane!