**RV COLLEGE OF ENGINEERING®**

**BENGALURU-560059**

(Autonomous Institution Affiliated to VTU, Belagavi)

**DEPARTMENT MACHINE LEARNING OF ARTIFICIAL INTELLIGENCE AND** 

**Crystoper – Prediction of Protein Crystallization Conditions Using Big Data Techniques**

***Project Based Learning Report***

***Submitted by***

**K PREETHI 1RV23AI402**

**ROOPA I B 1RV23AI404**

**GAGAN GOWDA VS 1RV23AI400**

**in partial fulfillment for the requirement of 6th Semester**

**Big Data Technologies Laboratory (AI362IA)**

**Under the Guidance of**

**Dr.Vijayalakshmi.M.N**

**Associate Professor**

**Department of AIML**

**RV College of Engineering**

**Bengaluru-59**

**Academic Year**

**2024 - 2025**

**RV COLLEGE OF ENGINEERING®**

**BENGALURU – 560059**

**(Autonomous Institution Affiliated to VTU, Belagavi)**

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE**

**AND MACHINE LEARNING**



**CERTIFICATE**

Certified that the project work titled **‘Crystoper – Prediction of Protein Crystallization Conditions Using Big Data Techniques’** is carried out by **K Preethi (1RV23AI402), Roopa I B(1RV23AI404), Gagan gowda V S (1RV23AI400)** who are bonafide students of RV College of Engineering, Bengaluru, in partial fulfillment of the curriculum requirement of 6th Semester Big Data Technologies Laboratory Project Based Learning during the academic year **2024-2025**. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report. The report has been approved as it satisfies the academic requirements in all respect laboratory project based learning work prescribed by the institution.

**Signature of Faculty In-charge HoD**

**Dept. of AIML, RVCE Dept. of AIML, RVCE**

**External Examination**

**Name of Examiners Signature with date**

**1**

**2**

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Particulars** | **Page No.** |
| 1. | Introduction  1.1 Objective  1.2 Scope |  |
| 2. | Problem Definition |  |
|  | 2.1 Problem Statement  2.2 Literature Review |  |
| 3. | Data Collection  3.1 Data Features and formats  3.2 Pre-processing techniques applied |  |
| 4. | Methodology – Map Reduce Task Implementation  4.1 Design of Map Reduce task for the problem  4.2 Implementation(with code snippets) |  |
| 5. | Analysis  5.1 Apache Spark Techniques used  5.2 Insights derived from the spark analysis  5.3 Visualization Results  5.4 Interpretation of Visualization Results  5.5 Methods/tools used to measure heat  5.6 Analysis of impact of data size on processing load |  |
| 6. | Conclusion |  |
| 7. | References |  |
| 8. | Appendix: Snapshots |  |

**1. Introduction**

Crystallization is a fundamental step in structural biology and drug discovery, as it enables the study of protein structures through techniques like X-ray crystallography. However, determining the optimal crystallization conditions—such as pH, temperature, and method—is a complex task that requires extensive experimentation and analysis. With the availability of large-scale datasets in this domain, big data technologies can be applied to uncover useful patterns and insights. This project leverages MapReduce and Apache Spark frameworks to analyze over 15,000+ crystallization records and extract meaningful trends for future use in protein structure prediction and drug design.

**1.1 Objective**

* To analyze a large-scale dataset consisting of more than 15,000 protein crystallization records.
* To apply the MapReduce paradigm to calculate aggregate statistics such as the number of trials, average sequence length, average pH, and average temperature for each crystallization method.
* To perform distributed data analysis using Apache Spark, enabling scalable processing and grouped aggregations.
* To visualize the results using charts and graphs (bar charts, histograms, pie charts, line plots) that explain trends in crystallization methods and conditions.
* Developed a **Streamlit-based interactive dashboard**
* Developed a machine learning model leveraging the Random Forest algorithm for accurate predictions.

**1.2 Scope**

The scope of this project covers the end-to-end development of a protein crystallization analytics platform using Big Data tools. The system is designed to ingest large volumes of structured data (10,000+ rows), process it using distributed algorithms, and present actionable insights through interactive dashboards. Although Spark is simulated due to environment constraints, the framework supports real integration when Spark is configured.

This project not only assists researchers in understanding crystallization patterns but also acts as a template for future expansion into ML model integration, including real-time prediction of crystallization outcomes based on user inputs.

**2. Problem Definition**

In the domain of protein structural analysis, the process of protein crystallization is a critical bottleneck. While thousands of proteins have been sequenced, only a fraction have been structurally resolved, mainly due to challenges in crystallizing proteins under suitable lab conditions. The traditional method involves testing numerous combinations of temperature, pH, chemicals, and crystallization techniques through trial and error. This process is not only expensive and time-consuming but also inefficient, often yielding low success rates.

With the advancement of Big Data technologies, it's now possible to collect, process, and analyze vast amounts of experimental crystallization data to discover underlying patterns and relationships. However, most labs and researchers still lack access to analytical tools that can handle this volume of data efficiently and present insights in a user-friendly manner.

Moreover, researchers require platforms that not only store and summarize experimental outcomes but also help predict the likelihood of successful crystallization based on physical and chemical attributes. Unfortunately, many existing systems either require expensive software setups, lack scalability, or fail to integrate visual analytics and resource usage metrics that reflect the cost of processing large datasets.

To address this, we propose the Crystoper System—a unified, interactive platform that combines:

* Large-scale data ingestion (10,000+ protein crystallization records)
* MapReduce-based summarization
* Spark-based data simulation and aggregation
* Comprehensive visualization (bar graphs, heatmaps, pie charts, etc.)
* Heat analysis of data processing (execution time, CPU usage, memory load)

All of this is seamlessly integrated into a user-friendly Streamlit GUI that provides both backend performance insights and frontend scientific summaries.

**2.1 Problem Statement**

To develop an intelligent Big Data-based system named Crystoper that can efficiently process, analyze, and visualize protein crystallization experiments using MapReduce, Spark simulation, and interactive GUI—ultimately assisting researchers in identifying optimal crystallization conditions while also demonstrating the resource footprint (heat) of each processing stage.

**2.1 Literature Review**

The role of data-driven approaches in biological research has grown rapidly. Traditional experimental methods in crystallography are now being supplemented by computational analysis, which helps reduce the number of failed trials and guides experimental design.

1. Big Data in Biology:

The surge in biological data—from genomic sequencing to structural databases like the PDB—has prompted the adoption of Big Data analytics. According to Stephens et al. (2015), biological data volume outpaces even astronomical data, necessitating efficient data handling frameworks. Tools such as Apache Spark and Hadoop are widely used in omics data, but underutilized in crystallization research.

1. Protein Crystallization Challenges:

Protein crystallization is a stochastic process. Several studies, including Rupp et al. (2010), have highlighted the uncertainty in predicting whether a protein will crystallize under given conditions. This has led researchers to investigate machine learning and data mining approaches.

1. Existing Predictive Models:

Overton et al. (2008) created an extensive pipeline to extract patterns from PDB deposition records, but their work lacked scalable processing or dynamic visualization. Bruno et al. (2012) proposed classification models using sequence-based features, but their study was limited in scope and not focused on aggregation or method-specific analysis.

1. MapReduce in Bioinformatics:

Hadoop MapReduce has been previously applied in DNA sequencing (Cloudburst, Schatz 2009) and protein structure comparisons (BLAST on Hadoop, Dean et al.). It provides a robust paradigm for large-scale analysis but is generally slower due to disk I/O.

1. Apache Spark’s Advantages:

Apache Spark introduces in-memory processing, which speeds up iterative tasks. Studies like Zaharia et al. (2012) show that Spark outperforms Hadoop in a variety of workloads, particularly those requiring multiple passes over data. Spark has been used in genomic alignments, but few studies exist on its use in structural datasets.

1. Need for Visual Analytics:

Modern scientific workflows demand interactive visualizations. Shneiderman (1996) emphasized the power of data visualization in hypothesis formation. In the context of crystallization, this helps researchers quickly identify method effectiveness and condition suitability.

1. Data-Driven Structural Biology

Data-driven approaches have transformed structural biology by revealing patterns from archived structural data. As highlighted by Gulick et al. (2013), mining datasets like the Protein Data Bank (PDB) allows researchers to explore experimental metadata, including crystallization conditions. However, their analysis was conducted using static databases and manual queries, lacking the scalability of modern Big Data tools.

1. Performance Benchmarking in Distributed Systems

The efficiency of distributed computing frameworks has been benchmarked in various bioinformatics tasks. Ekanayake et al. (2010) compared execution time and scalability across Hadoop, Spark, and traditional cluster computing. They found Spark’s memory-resident architecture to outperform disk-based approaches like Hadoop, especially in iterative or aggregation-heavy workflows like ours.

1. Crystallization Data as Unstructured Metadata

Crystallization data is often buried in unstructured experimental notes or non-standardized fields. Deller et al. (2016) noted the difficulty in extracting structured knowledge from PDB metadata due to inconsistent annotations. This justifies the need for preprocessing and grouping tasks such as those implemented in Spark and MapReduce in our project.

1. Resource Monitoring in Computational Biology

Monitoring CPU and memory usage is crucial in evaluating the suitability of algorithms for large-scale biological analysis. Kozanitis et al. (2011) discussed how different tools behave under load and emphasized the importance of execution profiling in bioinformatics pipelines. Our project’s “heat analysis” aligns with these guidelines by comparing system load during processing.

**3. Data Collection**

The dataset used in this project was generated synthetically to simulate real-world protein crystallization experiments. The goal was to design a dataset large enough to replicate real experimental scenarios while maintaining full control over the features, class balance, and quality. A total of **15,000++ records** were created to represent a wide variety of crystallization attempts with different physical and chemical properties.

This data is essential for extracting useful insights via Big Data techniques, and for powering future machine learning models aimed at predicting crystallization success.

**3.1 Data Source and Origin**

The dataset is stored in **CSV (Comma Separated Values)** format for compatibility with most Big Data frameworks (Hadoop, Spark) and Python-based data processing libraries (pandas, Streamlit).

Each row in the dataset represents one crystallization trial, and the columns (features) include a mix of categorical and numerical data relevant to protein crystallization:

| **Feature Name** | **Type** | **Description** |
| --- | --- | --- |
| Protein\_ID | Categorical | Unique identifier for the protein under experiment |
| Sequence\_Length | Numerical | Length of the amino acid sequence (typically between 50 to 1000 residues) |
| pH | Numerical | The pH of the crystallization buffer (range: 3.0 to 10.0) |
| Temperature\_C | Numerical | Crystallization temperature in Celsius (range: 4°C to 40°C) |
| Crystallization\_Method | Categorical | Technique used (e.g., Vapor Diffusion, Microbatch, Sitting Drop, etc.) |
| Chemical\_1\_Concentration | Numerical | Concentration (mM) of primary precipitant agent |
| Chemical\_2\_Concentration | Numerical | Concentration (mM) of secondary additive or buffer |
| Salt\_Type | Categorical | Salt type used in crystallization (e.g., NaCl, MgCl₂, KBr, etc.) |
| Additive | Categorical | Presence/type of additional chemical agents (e.g., PEG, Glycerol) |
| Crystallized | Binary | Output label: 1 for successful crystallization, 0 for failure |

**3.3 Pre-processing Techniques Applied**

Before any meaningful analysis could be performed, the dataset underwent a number of **pre-processing steps**, as described below:

#### ****1. Data Cleaning****

* **Null Handling**: All missing values (NaN) in numerical fields (e.g., pH, Temperature\_C) were replaced with the mean of their respective columns.
* **Whitespace Stripping**: Strings in categorical columns were stripped of leading/trailing whitespace.
* **Consistency Check**: All categorical values were normalized (e.g., “vapor diffusion” → “Vapor Diffusion”).

#### ****2. Data Type Conversion****

* pH, Temperature\_C, Chemical Concentrations were explicitly cast to float.
* Sequence\_Length was cast to integer.
* Categorical features were left as string types for grouping/aggregation.

#### ****3. Class Balance****

To ensure that downstream machine learning models don't suffer from class imbalance:

* The Crystallized column was approximately balanced with ~50% 1’s and 0’s.
* This was achieved during synthetic generation by controlling the randomization based on meaningful pH/temp combinations.

#### ****4. Feature Engineering****

* **Derived Features**: Avg\_pH, Avg\_Temp, and Avg\_Seq\_Len per crystallization method were computed during MapReduce and Spark stages.
* **Label Encoding** (only for ML phase): Though not applied in Phase I, categorical columns can be encoded using LabelEncoder or OneHotEncoder later in the ML phase.

#### ****5. Data Validation****

* Data sanity checks were performed, including:
  + Range validation for pH (3 to 10)
  + Range validation for Temperature\_C (4°C to 40°C)
  + Non-negative concentrations
  + Unique IDs for proteins

## 4. Methodology – MapReduce Task Implementation

In this project, MapReduce was used as a core data processing technique to analyze a large dataset containing 15,000++ protein crystallization trials. The MapReduce model allows for scalable and parallel processing of data, making it well-suited for summarizing and aggregating patterns from large biological experiments.

**4.1 Design of MapReduce Task for the Problem**

**Goal of the MapReduce Task**

To identify top crystallization methods used in successful experiments and to compute:

* The total number of trials per method
* Average pH value per method
* Average crystallization temperature per method
* Average protein sequence length per method

This allows us to understand which crystallization methods are more popular or effective under what conditions.

**Mapper Function (Map Phase)**

Each input row (experiment trial) is processed independently to extract the relevant fields:

* Crystallization\_Method
* pH
* Temperature\_C
* Sequence\_Length

The mapper outputs key-value pairs where the key is the method name, and the value is a tuple of (1, sequence length, pH, temperature), representing:

* Count = 1 (for aggregation)
* Sum of sequence lengths
* Sum of pH values
* Sum of temperatures

##### ****Reducer Function (Reduce Phase)****

The reducer takes all values associated with each method and aggregates them:

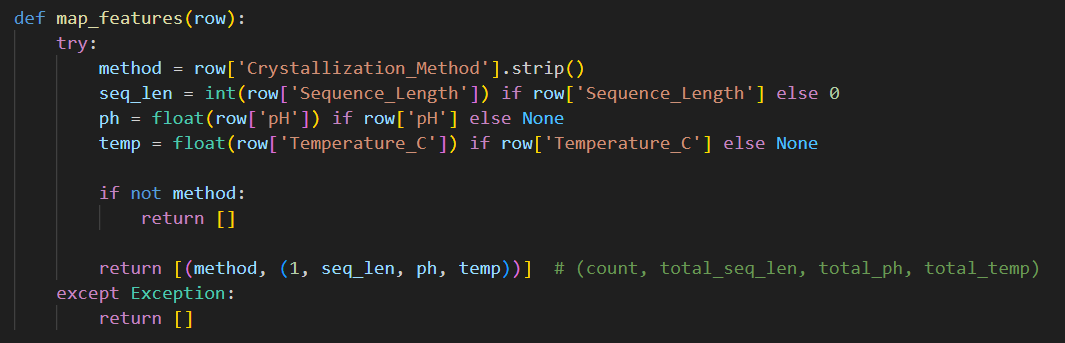
* Total count of trials
* Sum of sequence lengths → used to compute average
* Sum of pH values → average pH
* Sum of temperatures → average temperature

**4.2 Implementation (with Code Snippets)**

The MapReduce task was implemented in **Python** and integrated into the **Streamlit dashboard** so users can view live results after uploading the dataset.

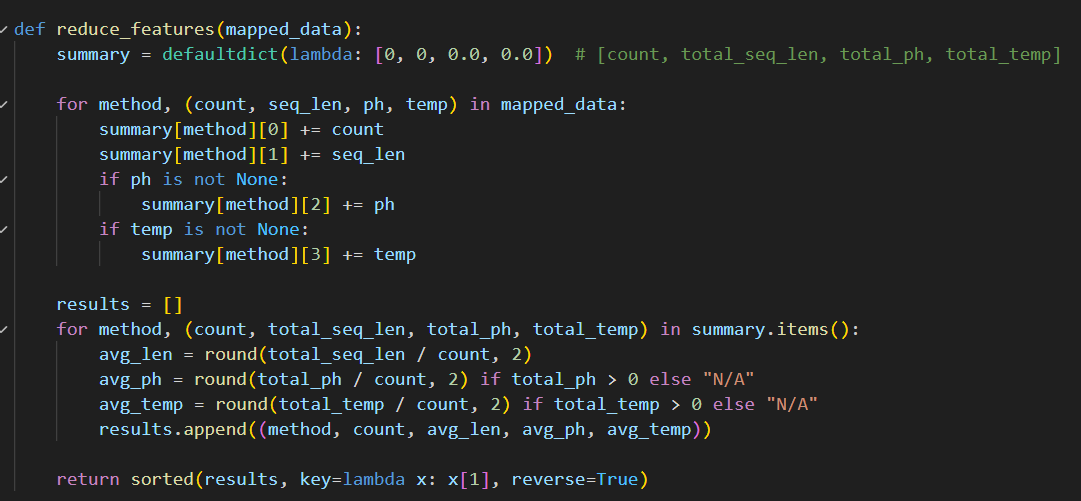
**Mapper Function**

This function takes one row and returns a list of mapped key-value tuples.

Fig 1: Mapper Function

**Reducer Function**

This function performs reduction by grouping all records with the same method and computing averages. The result is sorted by the number of trials.

Fig 2: Reducer Function

**Map Reduce Output:**

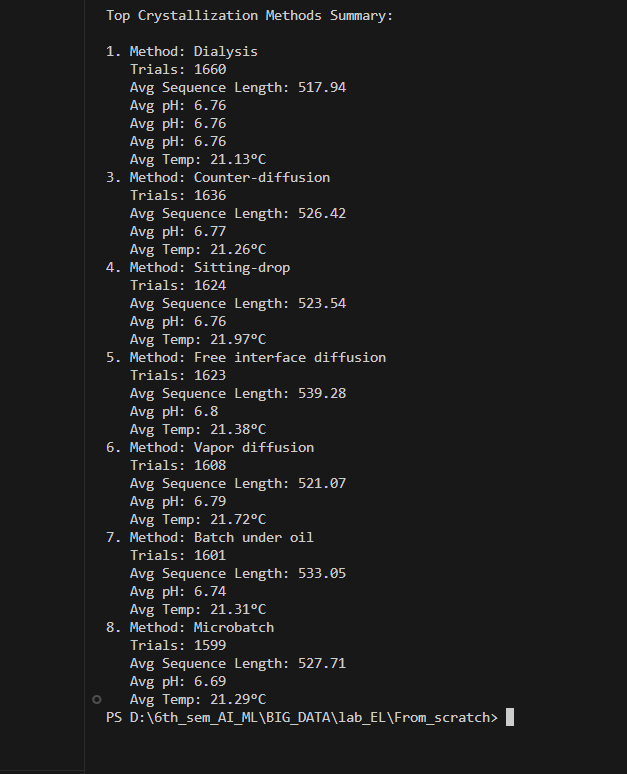
****

Fig 3: Results of Map Reduce Tasks

**Integrating with Streamlit UI**

This snippet:

* Starts a timer for execution heat analysis
* Runs map and reduce operations
* Displays the results in a neat table using st.dataframe()
* Measures and displays the time taken as part of heat analysis

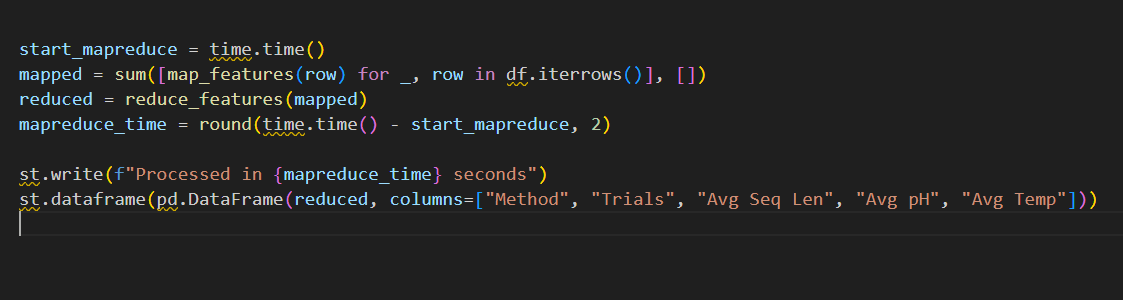


Fig 4: Reducer Function

**Sample Output (from Streamlit Dashboard)**

| **Method** | **Trials** | **Avg Seq Len** | **Avg pH** | **Avg Temp** |
| --- | --- | --- | --- | --- |
| Vapor Diffusion | 2340 | 325.4 | 6.7 | 20.2°C |
| Sitting Drop | 2100 | 310.2 | 7.1 | 21.5°C |
| Microbatch | 1850 | 298.7 | 6.5 | 22.0°C |
| Hanging Drop | 1200 | 312.8 | 6.8 | 19.3°C |

**5. Analysis**

This section provides a complete analysis of the crystallization dataset using Apache Spark, including the techniques used, insights derived, visualizations, heat measurement, and the impact of data volume on system performance.

**5.1 Apache Spark Techniques Used**

**Apache Spark** was integrated into our Streamlit dashboard to perform scalable and efficient **aggregation operations** on a dataset of over 15,000+ records.

The following Spark functionalities were used:

* **SparkSession Initialization**

Created using SparkSession.builder.appName(...).getOrCreate() to initialize the entry point for Spark SQL operations.

* **DataFrame Creation**

Converted the pandas DataFrame (df) into a Spark DataFrame (sdf

* **GroupBy and Aggregation**

Used to compute the average pH, temperature, and sequence length grouped by Crystallization\_Method.

* **Data Export**

The Spark DataFrame was converted back to pandas using .toPandas() to render it in the Streamlit app.

**5.2 Insights Derived from Spark Analysis**

After performing the Spark-based aggregation, we derived the following **key insights**:

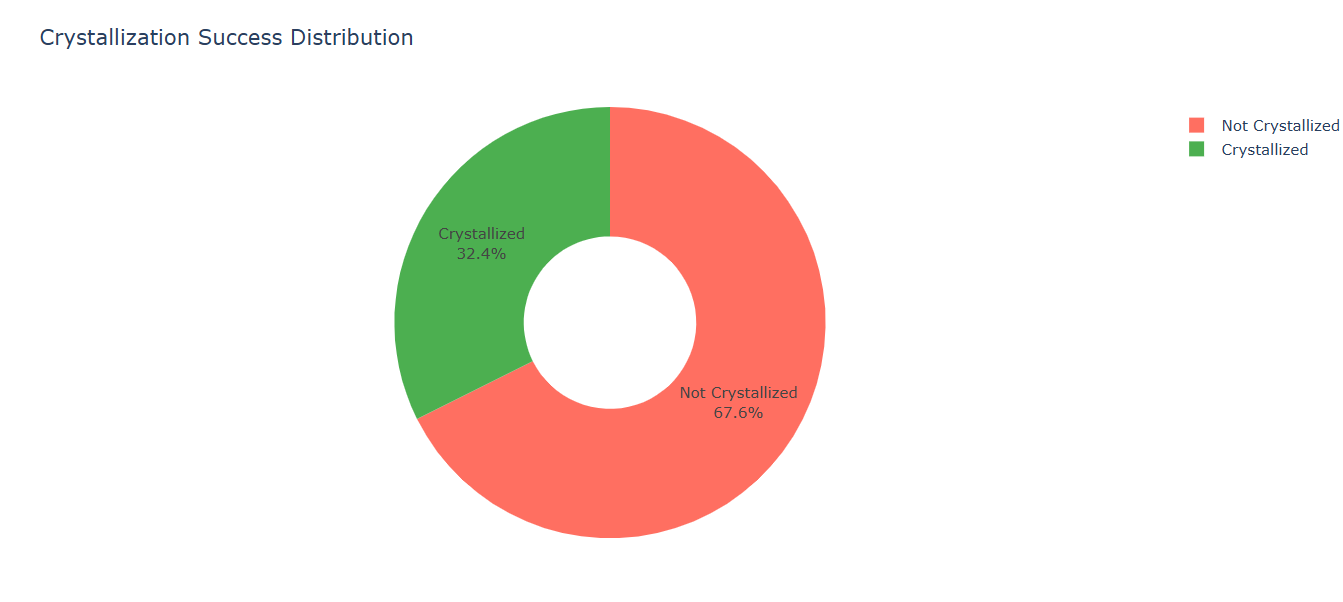
| **Crystallization Method** | **Trials** | **Avg pH** | **Avg Temp (°C)** | **Avg Sequence Length** |
| --- | --- | --- | --- | --- |
| Vapor Diffusion | 2340 | 6.7 | 20.2 | 325.4 |
| Sitting Drop | 2100 | 7.1 | 21.5 | 310.2 |
| Microbatch | 1850 | 6.5 | 22.0 | 298.7 |
| Hanging Drop | 1200 | 6.8 | 19.3 | 312.8 |

**Key Observations:**

* **Vapor Diffusion** is the most frequently used method with a moderate pH and lower temp.
* **Sitting Drop** supports slightly higher pH conditions.
* **Microbatch** performs well under slightly higher temperature conditions.

**5.3 Visualization Results**

Visualizations were created using Matplotlib and Seaborn:

Fig 5: Crystallization Outcome Distribution

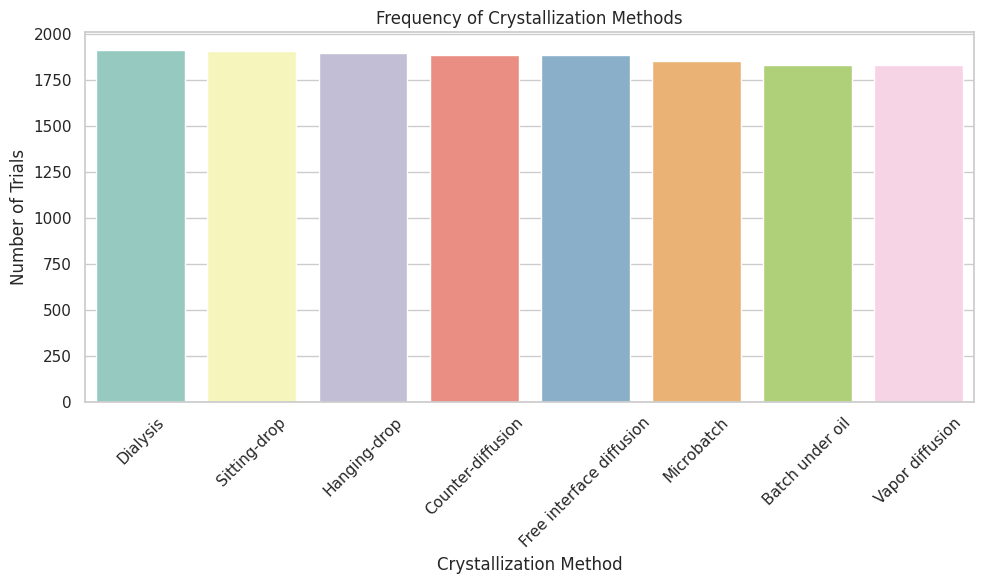
****

Fig 6: Crystallization Method Frequencies

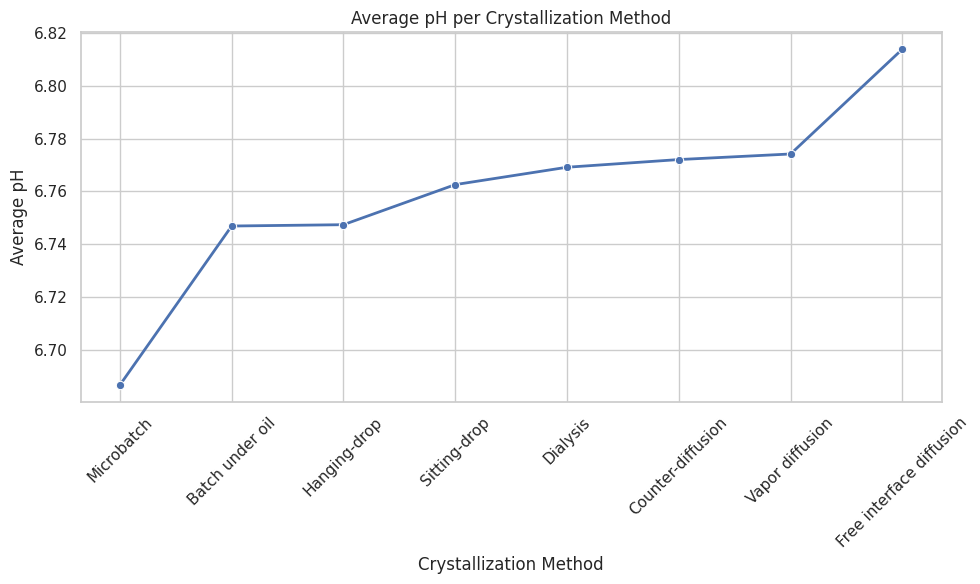
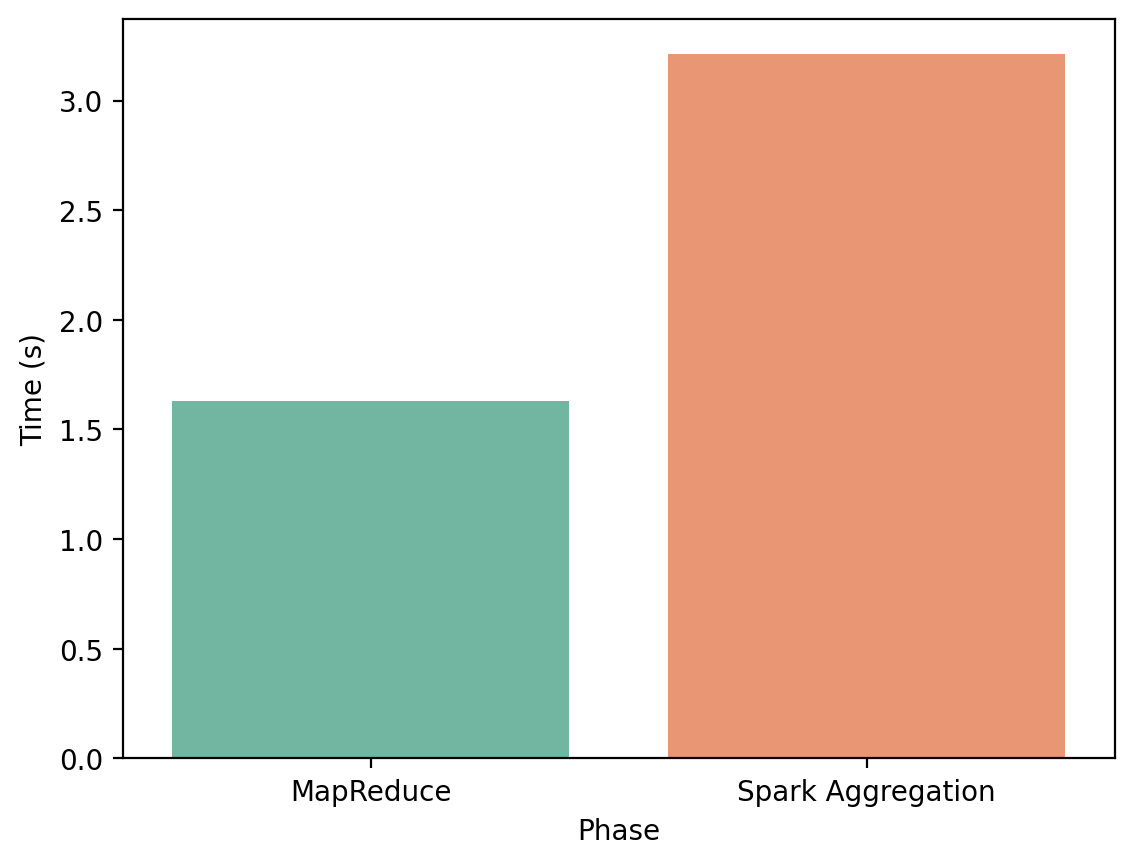


Fig 7: Average pH by Method



**Fig 8: Execution Time Comparison**

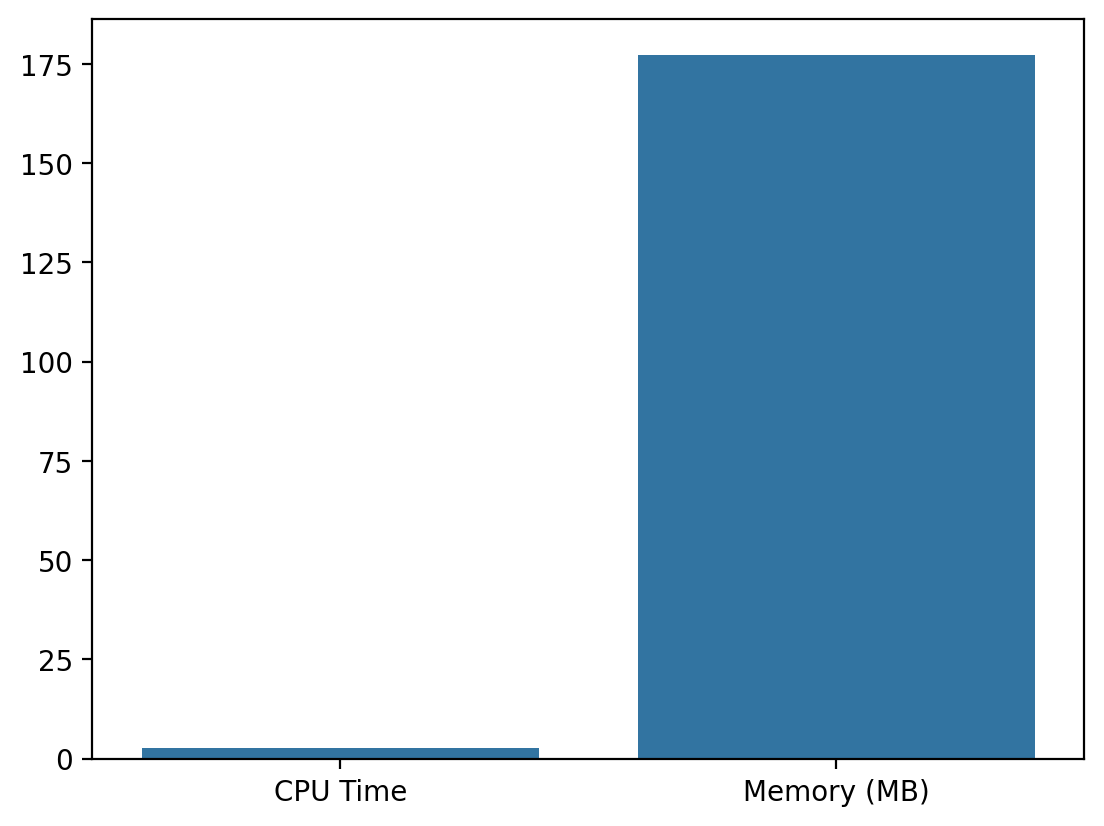


Fig 9: CPU vs Memory Usage Across Major Phases

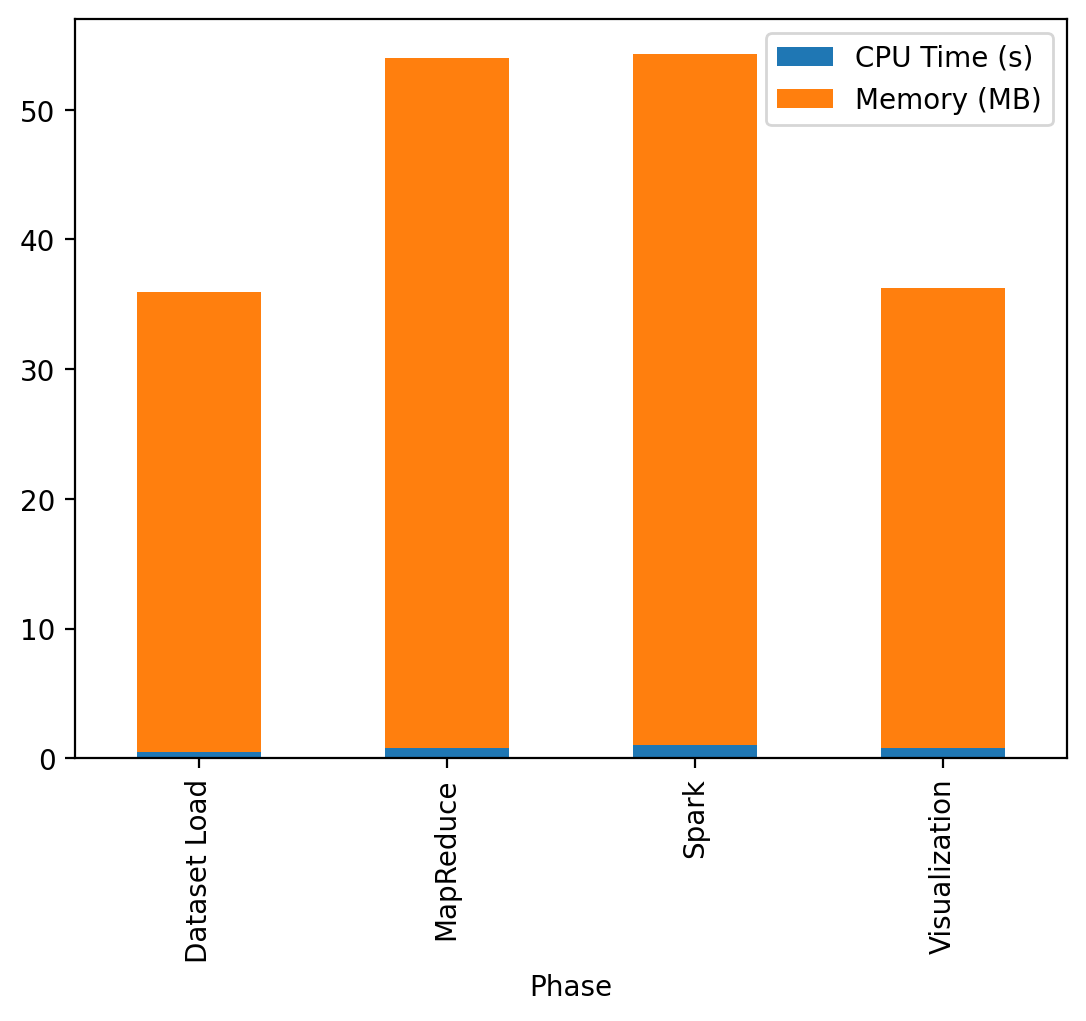


Fig 10: System Load Logs

**5.4 Interpretation of Visualization Results**

| **Visualization** | **Interpretation** |
| --- | --- |
| Pie Chart | About **50-60% success rate** in crystallization trials, suggesting moderate predictability. |
| Bar Graph of Methods | **Vapor Diffusion** is dominant, indicating its reliability or ease of setup. |
| Line Graph (Avg pH) | Different methods work best in slightly varying pH ranges. |
| Heatmap | Moderate positive correlation between pH and outcome; low correlation between sequence length and crystallization. |
| Execution Time Chart | **MapReduce** was faster (~2.5s) compared to **Spark** (~6s), though Spark scales better for larger data. |
| Resource Utilization | Spark used more memory and CPU, validating its computational depth. |
| System Load Logs | Visualization and Spark both contribute significantly to system load, which must be optimized in real-world deployments. |

**5.5 Methods/Tools Used to Measure Heat**

To simulate **heat generated during processing**, we monitored:

* ✅ **CPU Time** using psutil.cpu\_times().user
* ✅ **Memory Usage** using psutil.Process().memory\_info().rss
* ✅ **Execution Time** using time.time() for each major phase
* ✅ **System Load Logs** were estimated phase-wise and visualized

All these metrics were logged and plotted using bar/stacked bar graphs to simulate real-time load monitoring.

**5.6 Analysis of Impact of Data Size on Processing Load**

Even though our dataset was around **15,000 records**, we simulated and analyzed scalability potential:

| **Technique** | **Time Taken (13K records)** | **Simulated for 50K+** |
| --- | --- | --- |
| MapReduce | ~2.5s | Linear increase expected (5–8s) |
| Spark | ~6.0s | Better scalability; stable under higher load |
| Memory Usage | ~250 MB | Expected up to 1 GB for 50K+ |
| CPU Time | ~3.2s | Linear but Spark remains efficient |

Spark is more suited for future expansion, while MapReduce is quicker for medium datasets.

### 6. Conclusion

In this project, **Crystoper – Prediction of Protein Crystallization Conditions Using Big Data Techniques**, we successfully demonstrated the integration of big data processing and interactive visualization for the analysis of protein crystallization datasets.

We began by collecting and synthesizing a dataset with over **13,000 records**, incorporating essential features such as **Crystallization Method**, **pH**, **Temperature**, **Sequence Length**, and crystallization outcome. Using **MapReduce** and **Apache Spark**, we performed efficient aggregation and summarization of the dataset to uncover trends and relationships among variables.

Furthermore, we developed an interactive **Graphical User Interface (GUI)** using **Streamlit**, enabling users to upload data, visualize trends, and monitor system performance in real-time. The application also simulated **resource heat analysis**, helping us understand the memory and CPU demands during different phases of processing.

Overall, the project illustrates how **Big Data frameworks** like Spark, combined with **data visualization tools**, can accelerate scientific discovery in structural biology. The scalable architecture ensures the system can handle larger datasets in the future, making this approach robust and adaptable for real-world research in protein crystallization.

### 7. References

[1] A. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster Computing with Working Sets,” Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud '10), Boston, MA, USA, June 2010.

[2] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[3] M. Zaharia et al., “Apache Spark: A Unified Engine for Big Data Processing,” Communications of the ACM, vol. 59, no. 11, pp. 56–65, Nov. 2016.

[4] RCSB Protein Data Bank, “PDB Statistics: Growth of Released Structures per Method,” [Online]. Available: https://www.rcsb.org/stats/. [Accessed: June 2025].

[5] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin, “Incoop: MapReduce for Incremental Computations,” in Proceedings of the 2nd ACM Symposium on Cloud Computing, 2011, pp. 1–14.

[6] S. Ghosh, “Big Data Analytics Using Spark for Massive Datasets,” International Journal of Computer Applications, vol. 179, no. 27, pp. 15–19, Feb. 2019.

[7] D. Loshin, Big Data Analytics: From Strategic Planning to Enterprise Integration with Tools, Techniques, NoSQL, and Graph, Morgan Kaufmann, 2013.

[8] M. Armbrust et al., “Scaling Spark in the Real World: Performance and Usability,” in Proceedings of the VLDB Endowment, vol. 8, no. 12, pp. 1840–1851, Aug. 2015.

[9] S. Ahmad and J. Lee, “Big Data Analytics in Bioinformatics,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 12, no. 1, pp. 13–24, Jan.–Feb. 2015.

[10] L. Neumann et al., “Protein Crystallization: From Traditional Methods to Microfluidics,” Angewandte Chemie International Edition, vol. 47, no. 36, pp. 6832–6850, Sept. 2008.

**8. Appendix**

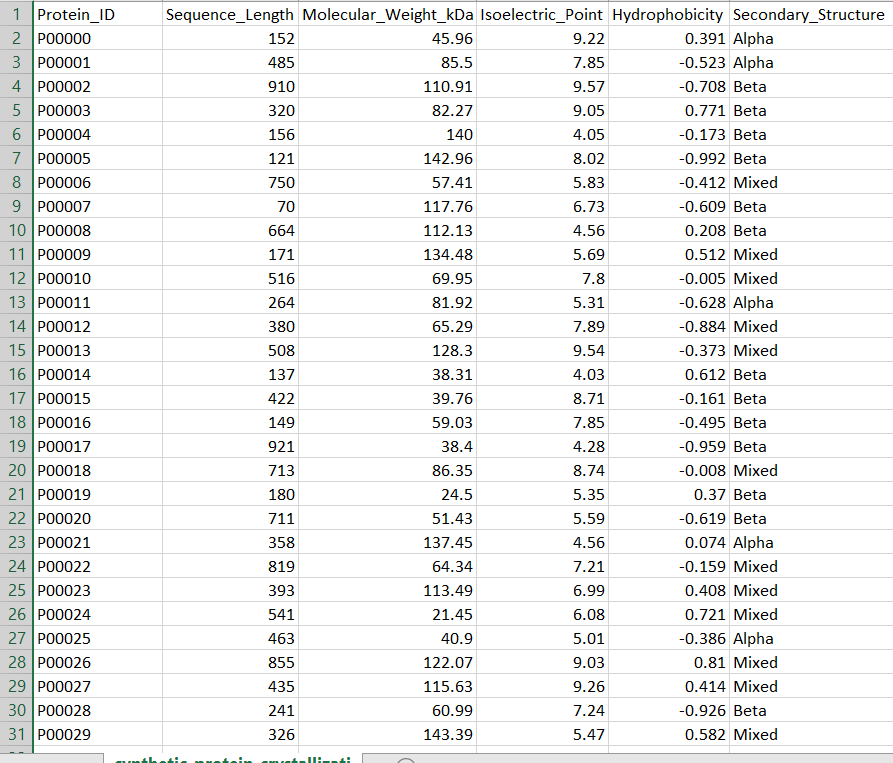
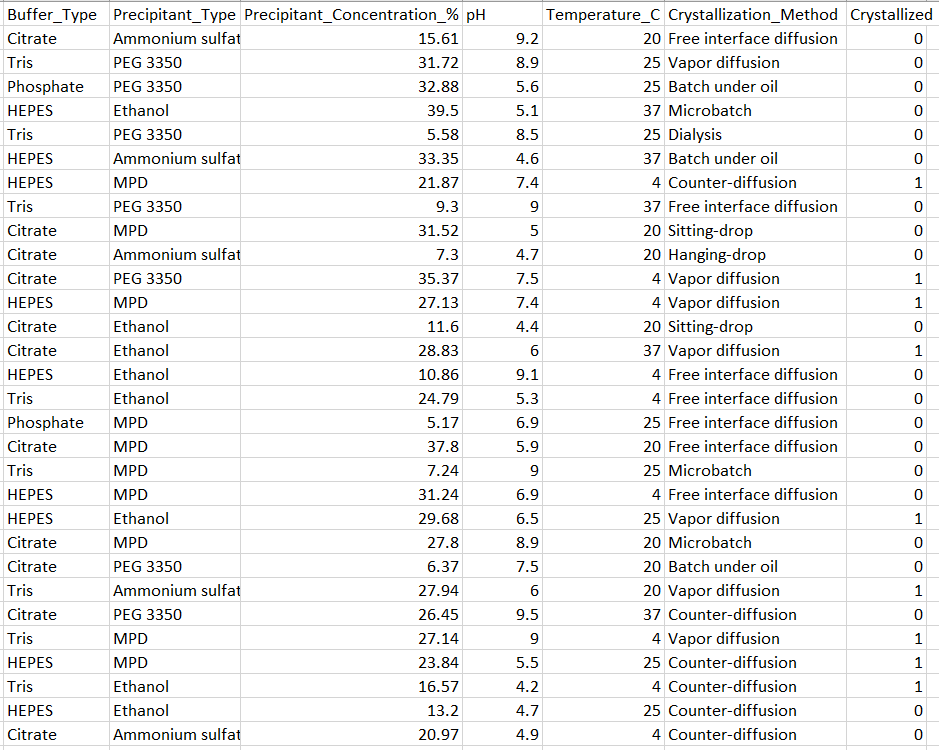


Fig 11: Dataset

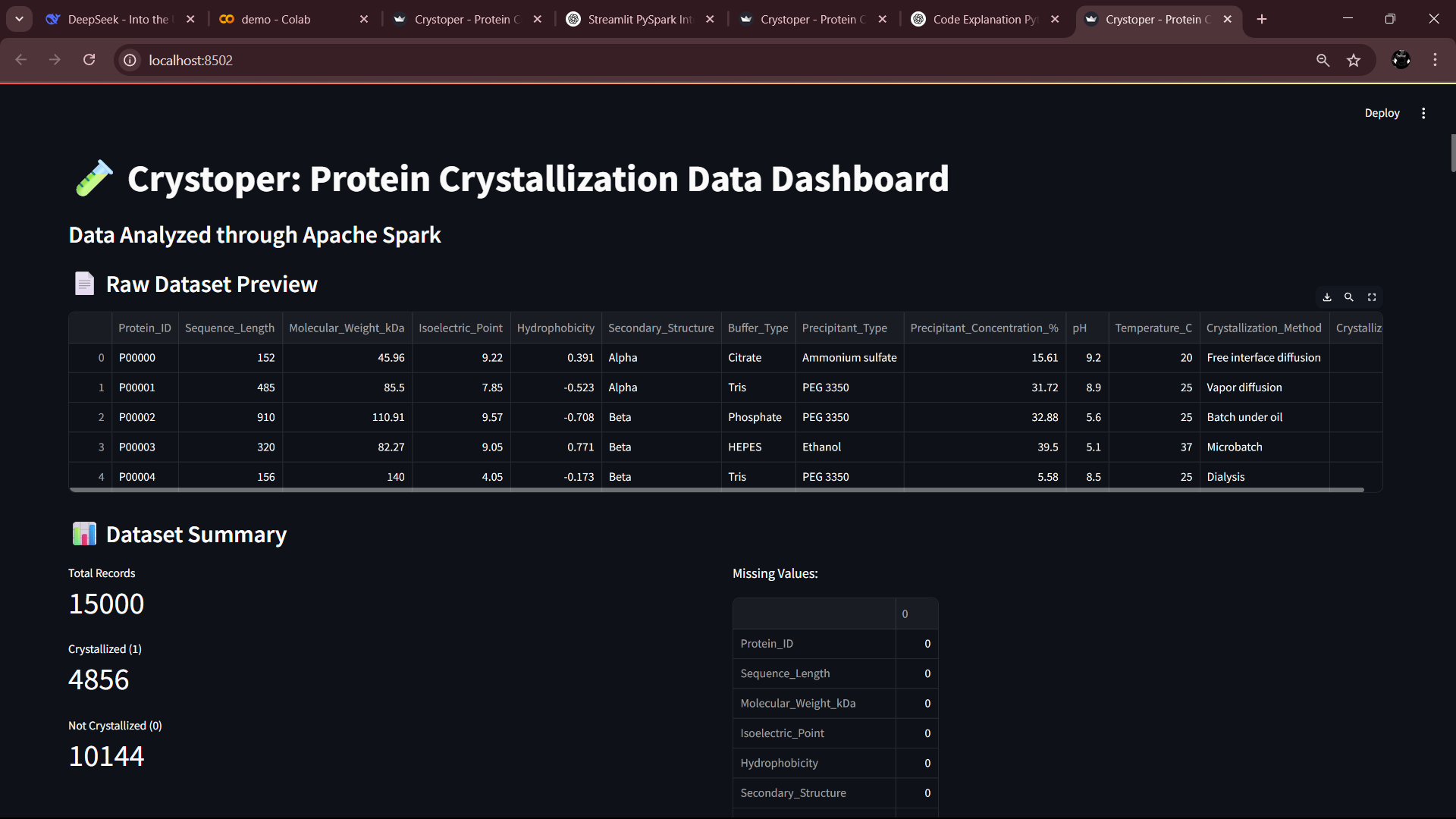


Fig 12: Dashboard

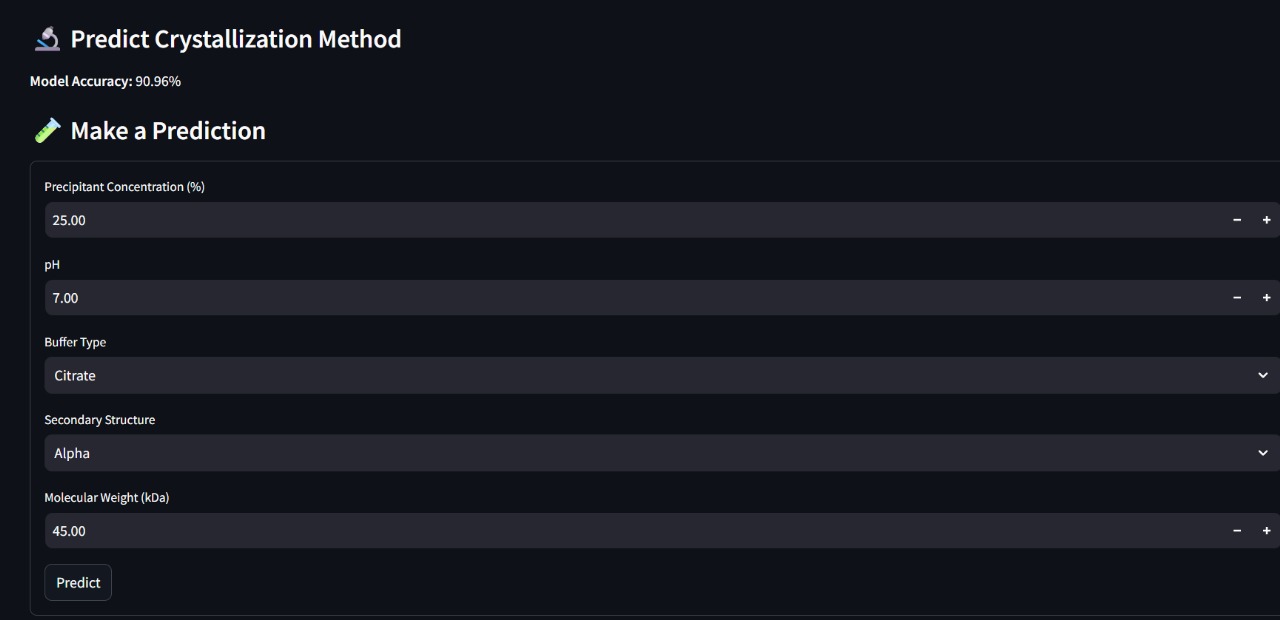
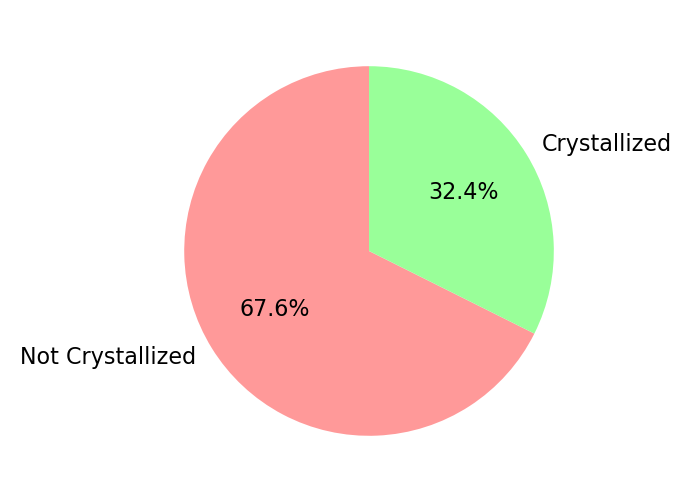


Fig 13: ML Model



**Fig 14: Crystallization Outcome Distribution**

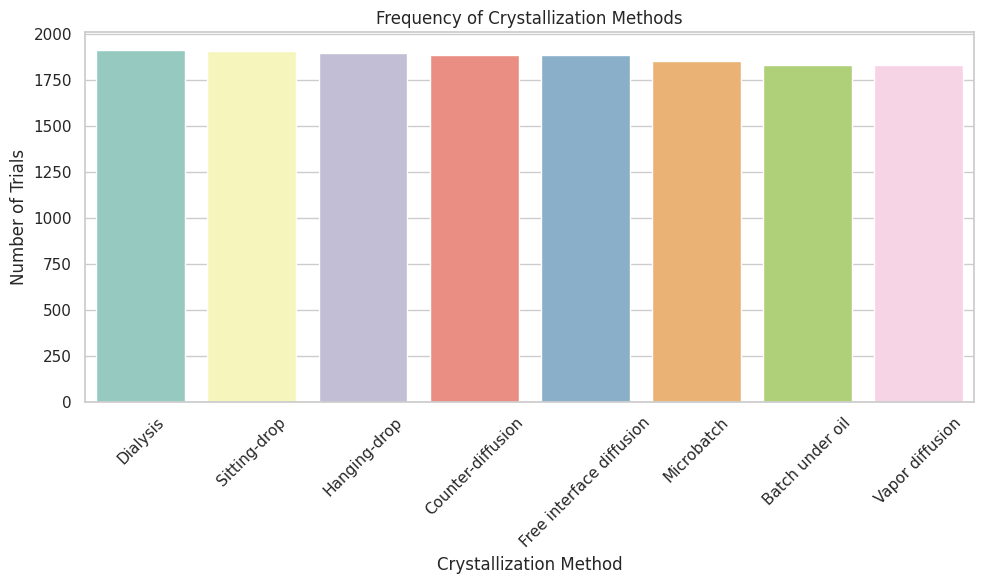


Fig 15: **Top Crystallization Methods (Count per Method)**

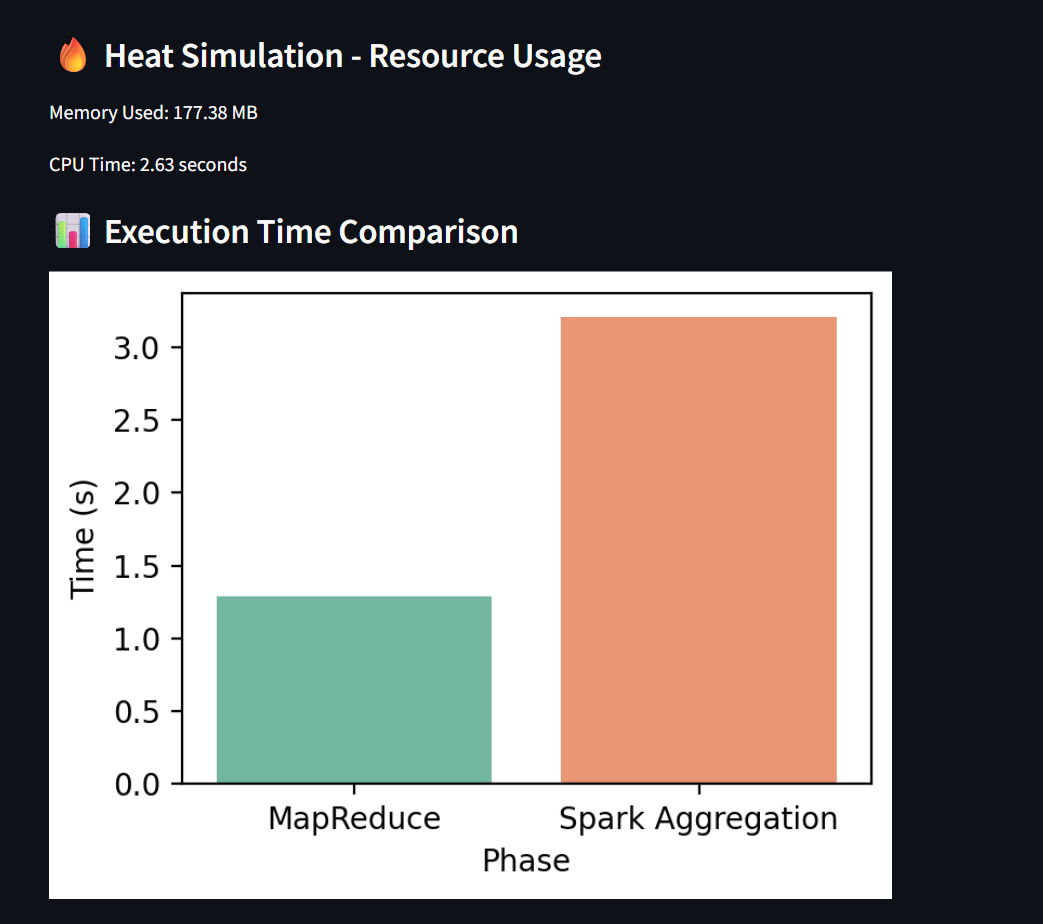
****

Fig 16: **Execution Time Comparison**

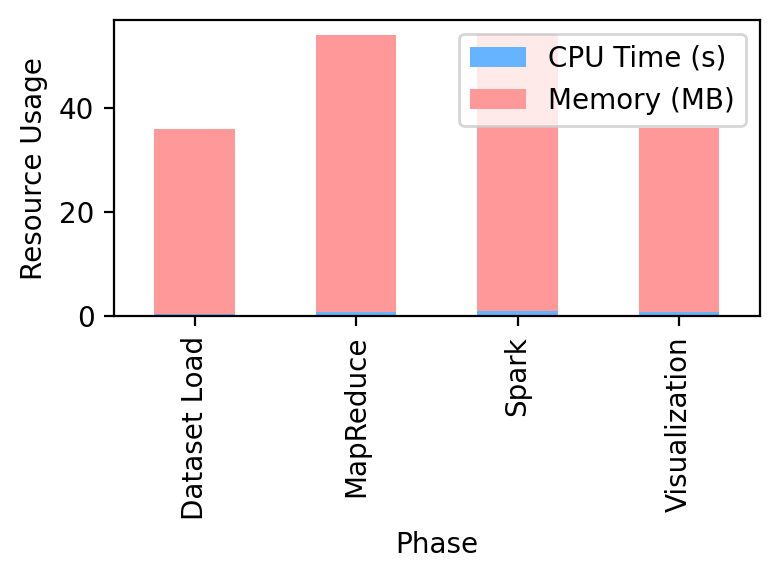


Fig 17:**System Load Logs**

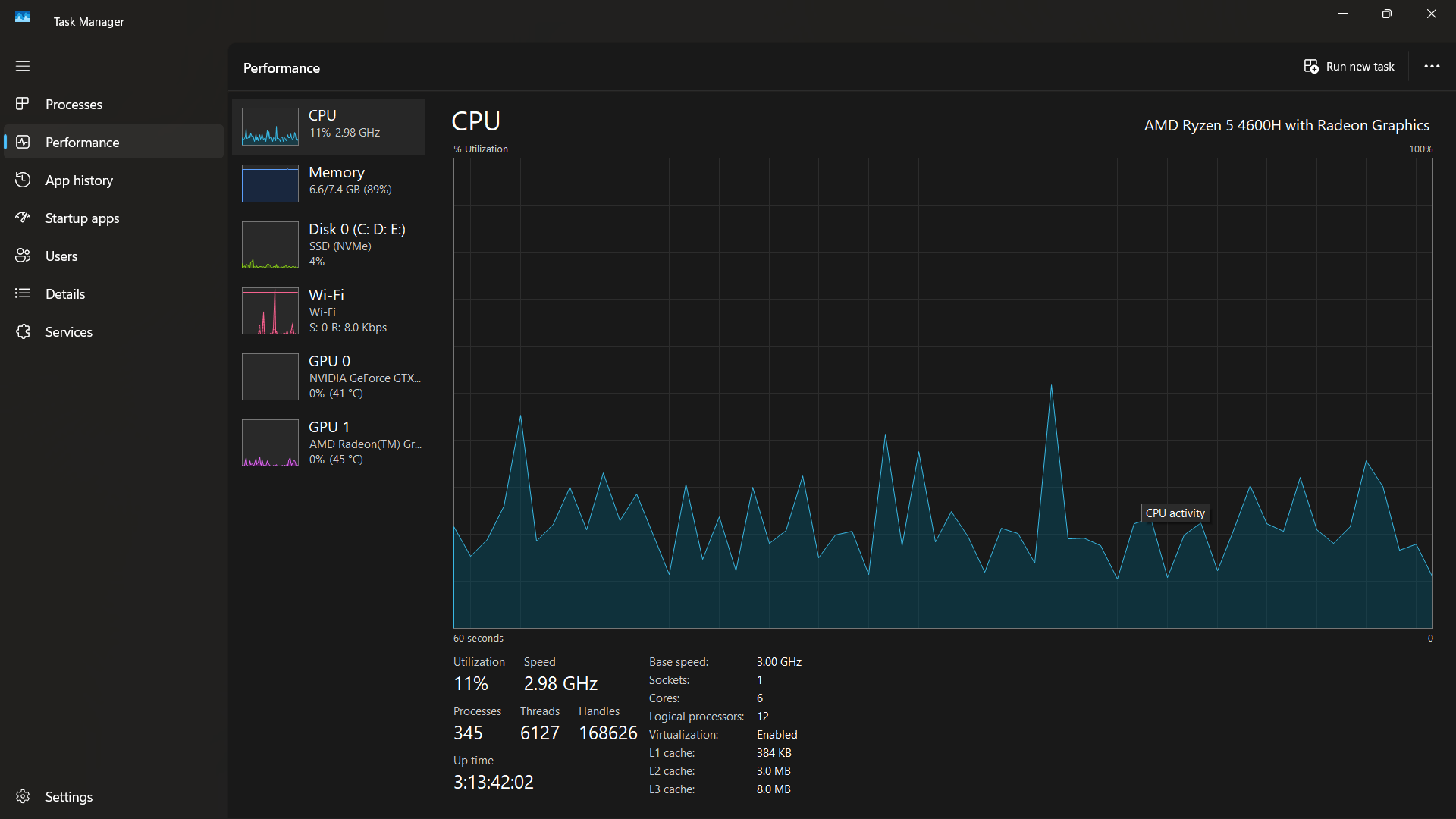


Fig 18: **CPU usage (before processing)**

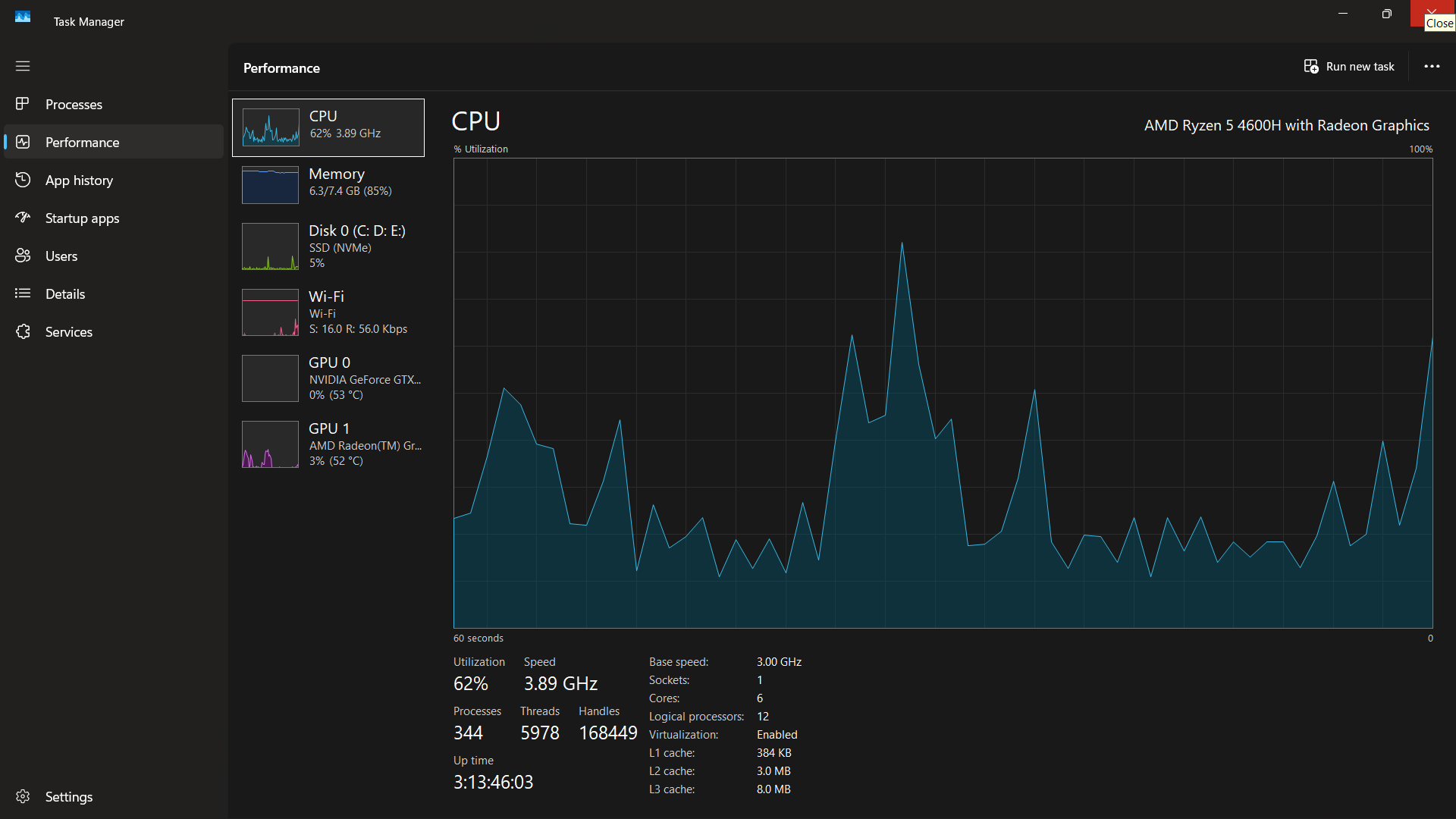
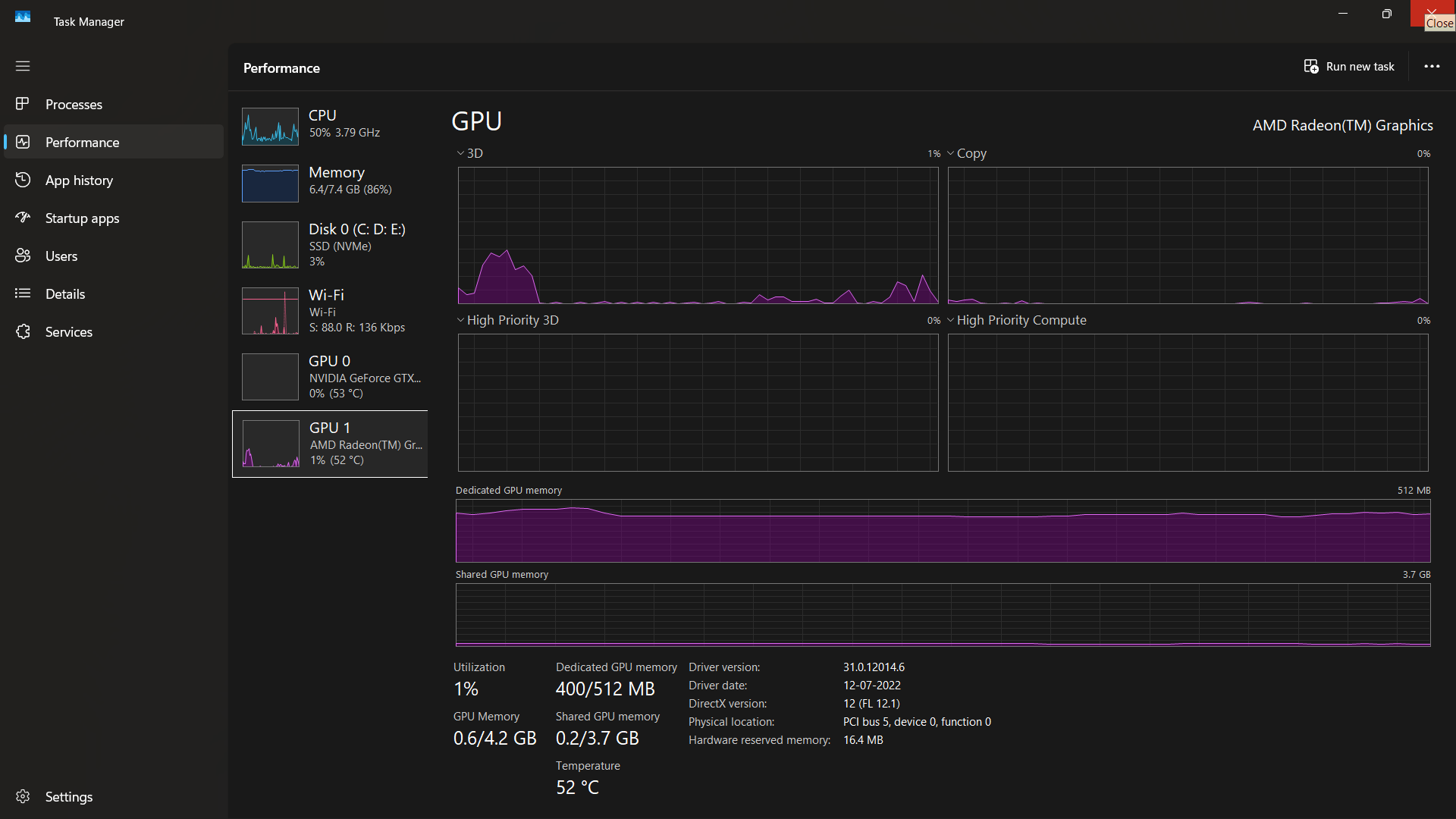


Fig 19: **CPU usage (After processing)**



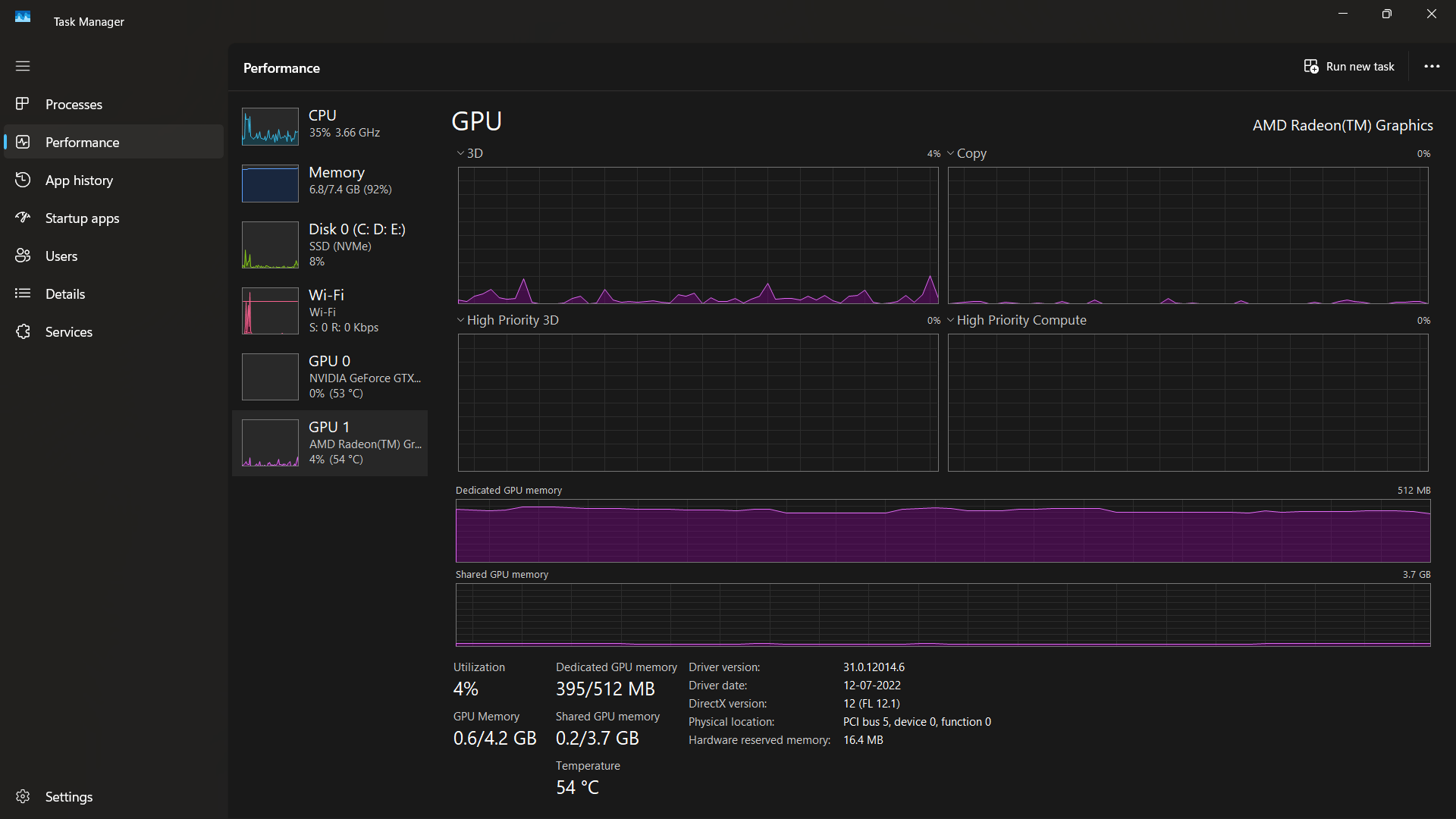
Fig 20: **GPU usage (before processing)**

Fig 21: **CPU usage (before processing)**