CER 4

Statistics and Continuous laws

Part 3

Date: 25/09/2024

Prosit 3

Statistics and Continuous Laws

Roles:

Facilitator: Maxime
Manager: Ethan
Secretary: Larry
Scribe: Andrew

1) Understanding the Topic and Clarifying

1.1) Keywords:

- Operating systems
- Predictive methods
- Key component of computer Operating system
- ~~Monitoring~~
- Descriptive statistics
- Metric
- ~~Resource management~~
- Optimize

1.2) Context:

An intern is tasked to analyze performance of an IT infrastructure to improve the system.

2) Needs Analysis:

2.1) Issues:

- How to manage and improve IT/ computer systems?
- How to improve key components of an optimal IT system?

2.2) Constraints:

- Analyzing data in 24hrs
- The 1-minute frequency intervals are too wide.
- Collecting data in real time
- Memory management

2.3) Deliverable:

- Report
- Python file/ Jupyter Notebook/ Excel
- (Graphical representation)
- Data analysis and conclusion from it

3) Generalization:

- Analysis and optimization of computer systems' key components
- Descriptive Statistics and continuous laws

4) Solution Tracks:

- Variables we have will not strongly correlate to dependent variable that systems performance.
- Positive correlation between the variables
- Independent variable (temp) and others are dependent variables.
- We will have to change components.
- Positive correlation between temperature and memory usage.
- Threshold limit for CPU usage and how to distribute it uniformly.

5) Developing the Action Plan:

1. **Define keywords.**
2. **Exercise Basket / WS.**
3. **Resource Analysis.**

4. Extract data
5. Check univariate statistics for each variable.
6. Visualize the data.
7. Clean data/ Detect bad data.
8. Get the correlation.
9. Name the most important variable.
10. Name alertness threshold.
11. Construct linear regression model/ equation.
12. Visualize the regression model.
13. Validation of assumptions
14. Conclusion

Keywords

Operating Systems (OS)

An Operating System (OS) is the software that manages the hardware and software resources of a computer. (“What is Operating System? Core Functions and Features”) It acts as an intermediary between the user or applications and the computer hardware. (“Resource management - Computer Science Wiki”) Key responsibilities of an OS include managing processes, memory, files, and peripheral devices, as well as providing security and a user interface.

Predictive Methods (in Operating Systems)

Predictive methods in operating systems refer to algorithms or techniques used to anticipate system needs or behaviours, usually to improve performance, optimize resource allocation, or prevent issues before they occur. These methods often rely on historical data and statistics to forecast future demands.

Examples in OS:

- **Predictive Paging**: Anticipates which pages of memory will be needed next and loads them into memory in advance to reduce page faults.
- **Predictive Disk Scheduling**: Uses patterns of disk access to predict and optimize future I/O operations.
- **Predictive Load Balancing**: Distributes tasks across processors based on predicted future load to improve efficiency.

Key Components of a Computer Operating System

1. **Kernel**: The core of the OS that manages CPU, memory, and device interactions. It operates in privileged mode and has full control over system resources.
2. **Process Management**: Manages the execution of processes, including scheduling, multitasking, and synchronization.
3. **Memory Management**: Allocates and manages system memory (RAM), ensuring efficient and secure memory usage through techniques like paging and virtual memory.

4. **File System Management**: Organizes and manages files on storage devices, including access control and directory management.
5. **Device Management**: Manages hardware devices such as keyboards, printers, and external storage, allowing applications to use them.
6. **User Interface (UI)**: Provides an interface (CLI or GUI) that allows users to interact with the system.

Descriptive Statistics

refers to methods used to summarize and describe the essential features of a dataset. (“Descriptive Statistics Made Easy: A Quick-Start Guide for Data Lovers”) These statistics give an overview of the data by providing measures of central tendency, variability, and distribution.

Metric

A **metric** is a standard of measurement used to assess and quantify performance, efficiency, or other relevant characteristics of a system or process. Metrics are often used in operating systems to monitor and evaluate system performance, resource usage, or system health.

Optimize

To **optimize** means to improve the efficiency, performance, or resource utilization of a system or process to achieve the best possible outcome within given constraints.

Examples of Optimization in OS:

- **CPU Scheduling Optimization**: Allocating CPU resources efficiently to minimize waiting time and maximize throughput.
- **Memory Optimization**: Reducing memory usage and avoiding memory leaks through better management of memory allocation and deallocation.
- **I/O Optimization**: Minimizing the time spent on I/O operations by using caching, buffering, or predictive scheduling techniques.

Law of Total Probability

The law of total probability is a fundamental rule in probability theory, relating marginal probabilities to conditional probabilities. ("Law of Total Probability - Vocab, Definition, and Must Know ... - Fiveable") It states that the total probability of an outcome can be expressed as a weighted average of conditional probabilities, where the weights are the probabilities of the conditioning events.

Mathematically, this can be represented as:

$$P(A) = \int_{-\infty}^{\infty} P(A|X = x)dF_X(x).$$

where P(A) is the total probability, P(A|X=x) is the conditional probability given X=x, and F_X(x) is the distribution function of X.

Law of the Unconscious Statistician

The law of the unconscious statistician (LOTUS) is a theorem that expresses the expected value of a function g(X) of a random variable X in terms of g and the probability distribution of X. This theorem is a generalization of the law of total probability and has applications in mathematical analysis and Lebesgue integration.

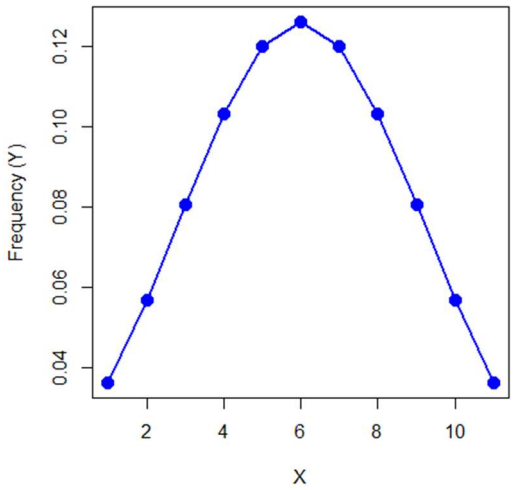
$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x)\,\mathrm{d}F_X(x).$$

Continuous Distributions

Continuous distributions, such as the normal distribution, play a crucial role in classical statistics. The Central Limit Theorem (CLT) states that the sample mean of a large random sample from a population will be approximately normally distributed, regardless of the underlying distribution of the population. This has important implications for statistical inference and hypothesis testing.

$$Y_i = \frac{0.398947}{3.16} \cdot e^{\frac{-(X_i-5)^2}{2 \cdot 10}}$$

Y_i is the height of the curve (normal density)
π (pi) is a constant = 3.14159... (R, use `pi`)
μ is the population mean
σ^2 is the population variance
σ is the square-root of the variance or the population standard deviation
e is the natural logarithm (R, use `exp()`)
X_i is the individual's value

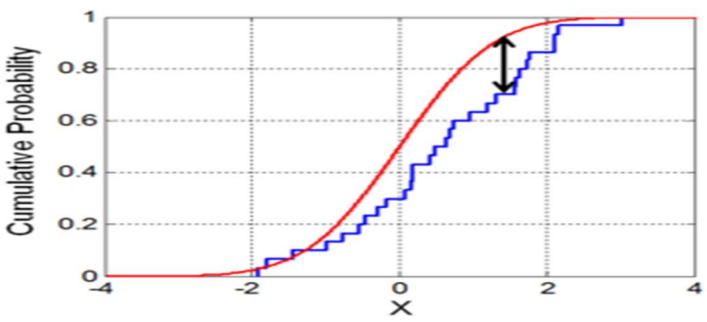


Key Takeaways

1. The law of total probability provides a framework for combining conditional probabilities to obtain total probabilities.
2. The law of the unconscious statistician generalizes the law of total probability to a broader context.
3. Continuous distributions, such as the normal distribution, are fundamental in classical statistics and are often assumed in statistical models.
4. The Central Limit Theorem (CLT) describes the convergence of sample means to a normal distribution, which has significant implications for statistical inference.

Kolmogorov–Smirnov test (**K–S test** or **KS test**) is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to test whether a sample came from a given reference probability distribution (one-sample K–S test), or to test whether two samples came from the same distribution.

$$F_n(x) = \frac{\text{number of (elements in the sample} \leq x)}{n} = \frac{1}{n}\sum_{i=1}^{n} 1_{(-\infty,x]}(X_i),$$



How an **Operating System (OS)** Works

An Operating System (OS) acts as an intermediary between computer hardware and the applications or users. It manages the hardware resources and provides services for programs, ensuring that all tasks are executed efficiently and securely. The OS handles various critical operations such as:

1. Process Management: Managing the creation, scheduling, and termination of processes.

2. Memory Management: Allocating and deallocating memory to processes and ensuring efficient use of RAM.

3. File System Management: Handling files on storage devices, managing permissions, and providing access to files.

4. Device Management: Managing input/output (I/O) devices like keyboards, printers, and external storage.

5. Security & Access Control: Protecting resources by controlling user access, authentication, and managing permissions.

6. User Interface: Providing a user interface (CLI or GUI) for users to interact with the system.

Key Mechanisms of an OS

1. Kernel:

The core part of the OS that directly manages hardware resources. It operates in privileged mode and provides essential services like CPU scheduling, memory management, and interrupt handling.

2. User Space and Kernel Space:

The OS divides memory into two main areas:

- Kernel Space: Runs the core parts of the OS and has direct access to hardware.

- User Space: Where user applications run. It provides protection by preventing applications from accessing hardware directly.

Process Management

A process is an instance of a running program. The OS handles process management through several mechanisms:

1. Process Creation and Termination:

- When a program is executed, the OS creates a process, assigns resources (memory, CPU time), and sets up a process control block (PCB) containing information like process ID, status, and CPU registers.

- The process can be terminated when it completes its task or if an error occurs.

2. CPU Scheduling:

- The OS uses CPU scheduling algorithms (e.g., Round-Robin, First-Come-First-Served) to decide which process gets the CPU next.

- Each process gets time on the CPU, and if interrupted (e.g., I/O operation), it is moved to a waiting state, and the next process takes over.

3. Context Switching:

- When the OS switches from one process to another, it saves the current state (registers, PC) of the running process and loads the state of the next process. This is known as context switching and introduces some overhead

4. Inter-process Communication (IPC):

- Multiple processes may need to communicate, and the OS provides mechanisms like message passing, pipes, or shared memory.

5. Thread Management:

- Modern operating systems support multithreading, where processes can have multiple threads of execution. The OS schedules and manages these threads for efficient execution.

Memory Management

Memory management in an OS ensures efficient use of the system's RAM and provides each process with the memory it needs while keeping processes isolated for security.

1. Memory Allocation:

- When a process is created, the OS allocates memory to it. (“Lecture 3: I/O and Processes - University of California, San Diego”) Memory

may be allocated dynamically (as needed) during the lifetime of the process.

2. Virtual Memory:

- Virtual memory extends physical RAM by using disk space (swap space). This allows the system to run larger applications or multiple processes without running out of memory.

- The OS uses paging or segmentation to divide memory into chunks (pages or segments) and maps virtual memory addresses to physical addresses.

3. Paging:

- Memory is divided into fixed-size pages. The OS keeps track of which pages belong to which processes and loads/unloads pages to and from disk as needed (this is known as swapping).

- A page table stores mappings of virtual pages to physical frames.

4. Segmentation:

- Memory is divided into variable-sized segments, where each segment corresponds to a logical division (like code, data, stack). Segments can grow or shrink dynamically.

5. Memory Protection:

- The OS ensures that one process cannot access the memory of another process. Techniques like address space isolation ensure each process operates within its own virtual memory space.

6. Demand Paging:

- Pages are loaded into memory only when they are needed, optimizing memory usage. If a page is not in memory, a page fault occurs, and the OS loads it from disk.

File System Management

The OS handles file operations, including reading, writing, creating, and deleting files. It also manages permissions and the directory structure.

1. File Systems:

- OS provides various file systems (e.g., FAT, NTFS, ext4), each with different methods of organizing and storing data on disks.

2. File Permissions:

- Security policies ensure proper access control. Each file has associated permissions (read, write, execute) assigned to users or groups.

Analysis Tools for OS

To understand and analyze how an OS manages processes, memory, and other resources, various tools are used:

1. Process Management Tools:

- `top`, `htop`, `ps` (Linux/Unix): Show running processes, CPU, and memory usage.

- Task Manager (Windows): Provides a graphical view of processes, memory, and CPU usage.

- Process Explorer (Windows): Offers deeper insights into processes, threads, and DLL usage.

2. Memory Management Tools:

- `free`, `vmstat` (Linux): Provide details on memory usage, swap space, and caching.

- Performance Monitor (Windows): Can be used to track memory usage over time.

- Valgrind (Linux): Helps detect memory leaks in programs.

3. Disk and File System Tools:

- `df`, `du` (Linux): Show disk space usage and available space.

- Disk Usage Analyzer (Windows, Linux): Provides a graphical view of disk usage.

4. Network and I/O Monitoring:

- `iostat`, `netstat` (Linux): Monitor input/output and network usage.

- Resource Monitor (Windows): Displays network, disk, memory, and CPU usage.

5. **Virtual Memory Tools**:

- `pmap` (Linux): Shows memory map of a process.

- Sysinternals tools (Windows): Can track memory paging and virtual memory usage.

Summary

To summarize, OS performs critical tasks in managing hardware and software resources, with process and memory management being two core functions. Processes are managed using scheduling and communication mechanisms, while memory is optimized through virtual memory and paging techniques. A variety of tools are available to analyse how the OS handles these responsibilities, providing insights into system performance, memory usage, and process handling.

Resource Analysis

Data-driven alertness thresholds refer to the use of data, often in real-time, to set specific alert levels or thresholds in various contexts, typically for monitoring systems, safety protocols, health, or operational performance. These thresholds help identify when a situation needs attention based on the data being collected.

Example:
Operational Alert Thresholds in Systems Monitoring

In IT and operations, alert thresholds are commonly used to ensure the health of systems. Performance metrics such as CPU usage, memory consumption, network latency, or response time are constantly monitored.

Example:

- A server's CPU usage exceeds 90% for more than 5 minutes, which triggers an alert to the operations team.
- Real-time dashboards track system metrics, and thresholds are set based on historical data to identify anomalies.

Tools:

- Cloud monitoring tools like AWS CloudWatch, Datadog, or Prometheus.
- Anomaly detection algorithms for dynamic thresholding based on trends or seasonality.
-

Data extraction

Data types

```
RangeIndex: 10080 entries, 0 to 10079
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Time            10080 non-null  object
 1   CPU_Usage       10080 non-null  float64
 2   Memory_Usage    10080 non-null  float64
 3   Network_Usage   10080 non-null  float64
 4   Temperature     10080 non-null  float64
dtypes: float64(4), object(1)
memory usage: 393.9+ KB
```

Descriptive statistics

	count	mean	std	min	25%
CPU_Usage	10080.0	41.407736	20.593190	10.000000	25.633834
Memory_Usage	10080.0	7.199340	3.217598	2.000000	4.788939
Network_Usage	10080.0	115.010777	53.874221	25.000000	74.553980
Temperature	10080.0	41.365703	6.514586	27.796231	36.243500

	50%	75%	max
CPU_Usage	39.182779	55.393982	100.000000
Memory_Usage	7.138533	9.042282	16.000000
Network_Usage	108.463094	159.780990	200.000000
Temperature	40.806542	46.803793	57.858396

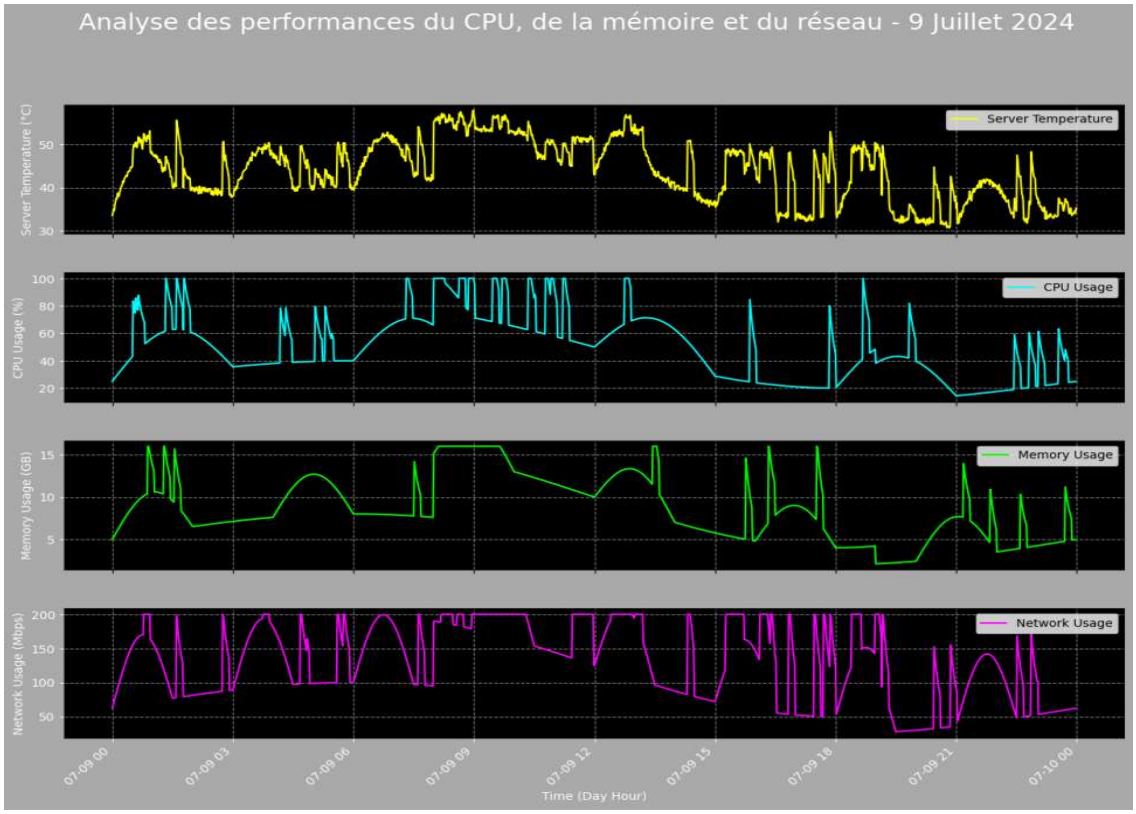
From this data we can, identify the alertness thresholds. Which is set to:

- CPU_Usage: 80%
- Memory_Usage:8
- Network_Usage:120
- Temperature: 40°

Data sample

2024-07-15 07:21:00	70.678897	7.814574	200.000000	53.282242
2024-07-15 07:22:00	70.738661	7.810017	191.046339	51.905758
2024-07-15 07:23:00	70.788505	7.805406	181.349996	50.482354
2024-07-15 07:24:00	70.828407	14.200741	172.219911	50.318030
2024-07-15 07:25:00	70.858348	13.586983	163.603472	50.508879
2024-07-15 07:26:00	70.878313	13.031130	111.307969	44.294922
2024-07-15 07:27:00	70.888288	12.527665	107.783205	44.146256
2024-07-15 07:28:00	70.888264	12.071600	104.245042	42.921138
2024-07-15 07:29:00	70.878234	11.658418	100.697719	43.204925
2024-07-15 07:30:00	70.858193	11.284033	97.145482	44.340760

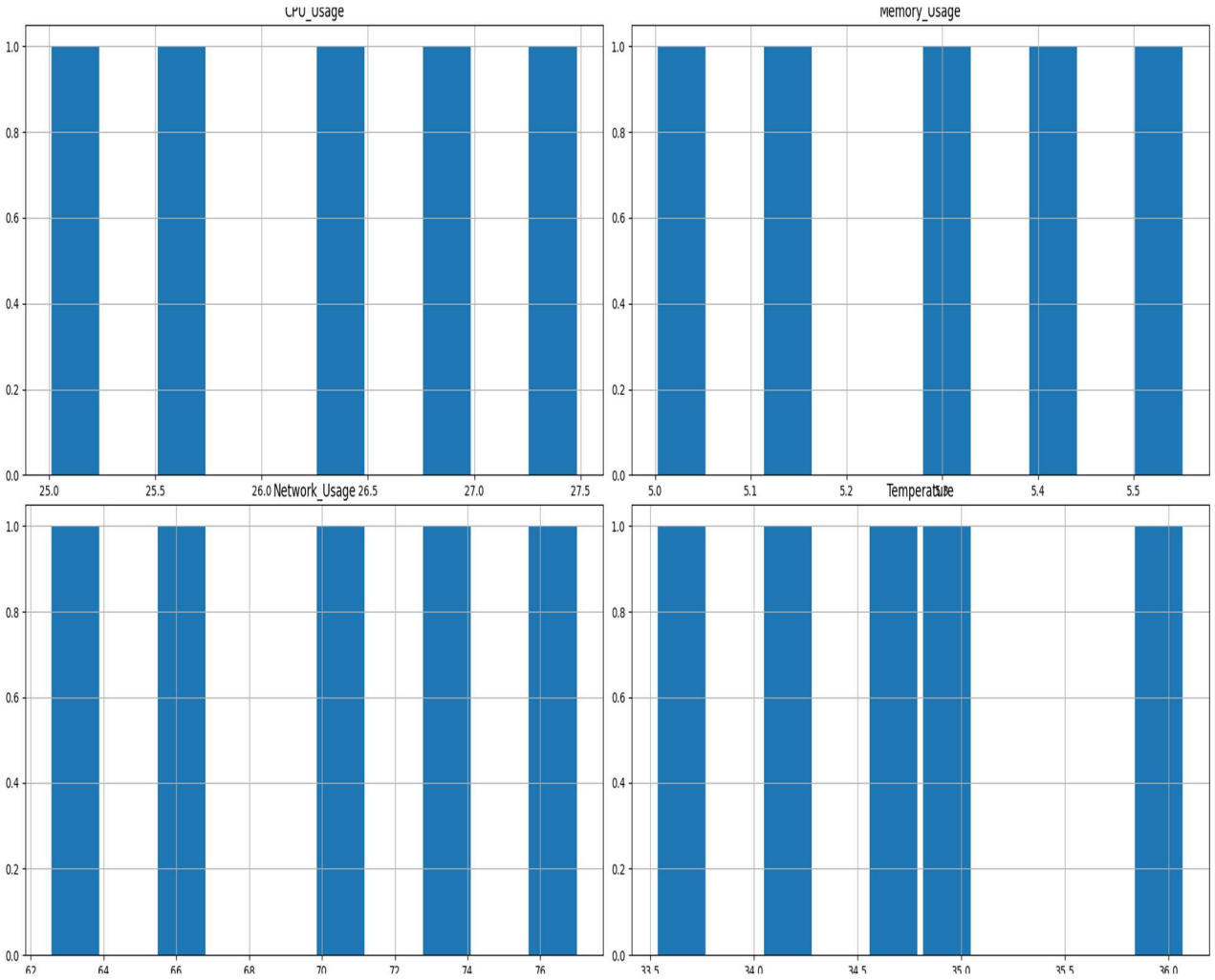
Solution



The image above shows the initial visual representation for each variable against time. From this, we could extrapolate various analysis and predictions.

Bar chart

Visual representation gotten for each variable, i.e. CPU, memory, network and temperature.



Anomalies

The search of abnormal data for each variable:

```
Q1 (25th percentile): 25.633833602687012
Q3 (75th percentile): 55.3939820930785
IQR: 29.760148490391487
Lower bound: -19.006389132900217
Upper bound: 100.03420482866574
Number of anomalies: 0
Anomalies:
Series([], Name: CPU_Usage, dtype: float64)
```

In other cases, there was no anomaly found apart from memory_usage:

```
Q1 (25th percentile): 4.78893883136604
Q3 (75th percentile): 9.042281754658633
IQR: 4.253342923292593 4
Lower bound: -1.591075553572850 7
Upper bound: 15.42229613959752 4
Number of anomalies: 175
Anomalies:
53      16.000000
54      16.000000
55      15.711870
77      16.000000
78      16.000000
79      15.471949
93      15.701582
483     15.533042
484     15.644740
485     15.755997
486     15.866726
```

This could've been caused by memory leak caused by poor programs i.e. allocates memory but does not properly release it after use, this could lead to the crashing or poor performance of the system

Correlation

Here is a graph showing the correlation between each variable against temperature.

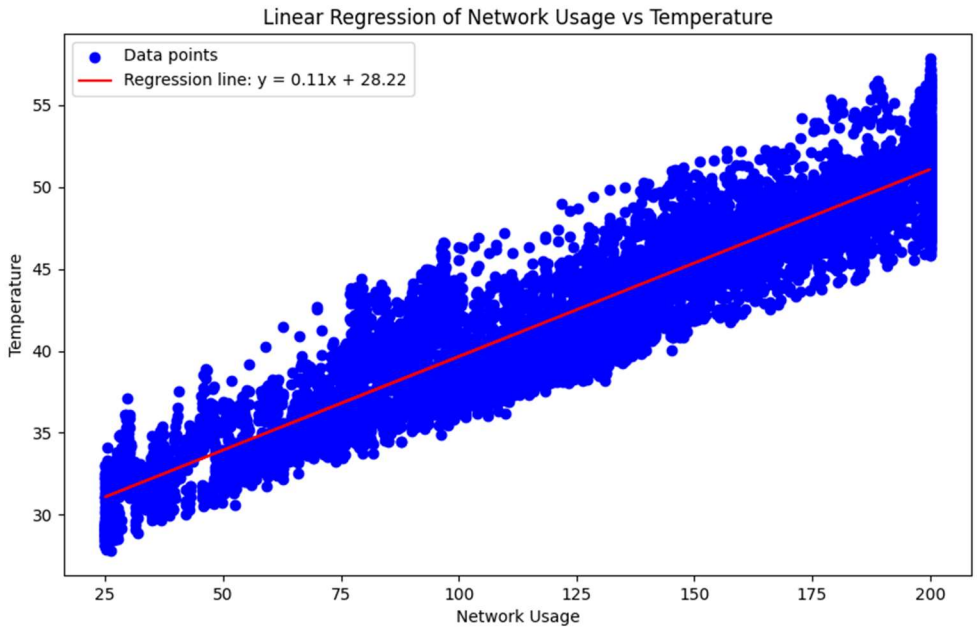
```
28  x= df['CPU_Usage'];
29  y= df['Temperature'];
30  corr = np.corrcoef(x,y);
31  print(corr[0,1])
```

```
Shell
>>> %Run -c $EDITOR_CONTENT
  0.5978263837900479
```

as observed above, the correlation is positive i.e. direct relationship between the two variables where they move in the same direction.

The biggest / strongest correlation observed was network_Usage against temperature which was 0.944877948. **Thus, implying as the network usage increase so does the temperature so if we want to optimise the computer infrastructure, we'd have to reduce network usage the most.**

Linear regression model



The above image is the linear regression model of Network_Usage vs Temperature. As explained before, we can observe that the network usage directly impacts the temperature of the server.

Conclusion

In conclusion, as shown and proven above, the most important factor or variable would be network usage i.e. if we could reduce it, this could directly impact the CPU such as the memory usage and therefore the temperature. This could lead to a more optimised IT infrastructure.

Data-driven alertness thresholds can help to optimize performance, prevent critical failures, and ensure timely interventions and could act as an added bonus security measure.