**You said:**

hello

**ChatGPT said:**

Hello there! How can I help you today?

2/2

**You said:**

i want to train a AI model which can process CPU logs. let me first give you what the dataset would look like.

**ChatGPT said:**

That sounds like a great project! Go ahead and share a sample of your dataset, and we can discuss how to preprocess it and train an AI model for CPU log analysis.

**You said:**

Date Time Core Clocks (avg) [MHz] Ring/LLC Clock [MHz] Core Usage (avg) [%] Core Temperatures (avg) [°C] CPU Package [°C] CPU Package Power [W] 24.3.2025 49:42.2 1820.5 1596.1 6.5 43 48 13.583 24.3.2025 49:43.3 1733.3 1296.8 6.3 43 48 14.829 24.3.2025 49:44.1 1438.1 798 4.3 43 48 9.753 24.3.2025 49:45.3 1513 798 3.8 42 48 11.077 24.3.2025 49:46.1 1758.2 1496.3 3.3 43 47 8.681 this is the dataset. we can of course add more features if required since I am using hwinfo for generating all the data.

**ChatGPT said:**

Thanks for sharing the dataset! This looks like a great starting point for analyzing CPU logs. You have several useful features such as:

* **Core Clocks (avg) [MHz]**
* **Ring/LLC Clock [MHz]**
* **Core Usage (avg) [%]**
* **Core Temperatures (avg) [°C]**
* **CPU Package [°C]**
* **CPU Package Power [W]**
* **Date and Time** (which can help with time-series analysis)

To train a model for processing CPU logs, we could focus on a variety of tasks, such as:

1. **Anomaly detection**: Identifying unusual CPU behavior (e.g., high temperature or power usage).
2. **Prediction**: Predicting future CPU behaviors based on historical data (e.g., predicting CPU temperature or power consumption).
3. **Classification**: Categorizing CPU states (e.g., normal, high load, overheating, etc.).

**Steps for preparing the dataset:**

1. **Feature Engineering**:
   * Convert the Date and Time into a more usable format (e.g., Unix timestamp or breakdown of hours, minutes, and seconds).
   * You might want to calculate rolling averages or statistical features (e.g., max, min, mean) over time windows to capture trends.
2. **Data Preprocessing**:
   * Check for missing values and handle them (e.g., imputation or removal).
   * Normalize/scale numerical features (e.g., temperature, clock speed) to ensure uniformity.
   * Ensure time consistency if you're dealing with time-series data.
3. **Data Labeling** (if necessary):
   * If you're doing supervised learning, label your data based on certain criteria (e.g., labels like “high load,” “normal,” or “overheating” for classification).
4. **Modeling**:
   * If you're predicting future behavior, you can use **time-series models** like **ARIMA**, **LSTMs**, or **GRUs**.
   * For anomaly detection, unsupervised models like **Isolation Forest** or **Autoencoders** could work well.
   * For classification tasks, **Random Forest**, **SVM**, or **Neural Networks** might be suitable.

**Additional Features (Optional):**

You can always add more features for better prediction or classification. Some potential features could include:

* **Load Average**: Over a set time window (e.g., 1 minute, 5 minutes).
* **CPU Frequency Range**: Min and max frequencies over a time window.
* **CPU Frequency Stability**: Standard deviation of core clock speeds.
* **Power Consumption Change**: Rate of change of power consumption over time.
* **Temperature Delta**: Rate of temperature increase/decrease.

What specific task are you trying to achieve with this model? That will help guide how we proceed with feature engineering and model selection.

**You said:**

now the thing i am looking for is training a decision tree based model. it should be able to process the logs and predict whether the CPU is thermal throttling, power limited for now. how can we do that?

**ChatGPT said:**

Great! If you're looking to train a **Decision Tree** model that predicts whether the CPU is **thermal throttling** or **power limited**, we can break down the process into a few key steps:

**1. Data Labeling:**

Before training a model, you'll need to label the data. To predict **thermal throttling** and **power limitations**, you'll need to define conditions for these two states. Typically:

* **Thermal Throttling**: This occurs when the CPU temperature exceeds a certain threshold (e.g., around 90-100°C). When thermal throttling happens, the CPU will slow down to reduce heat.
* **Power Limited**: This occurs when the CPU reaches a power consumption limit set by the system. It happens when the CPU can't get more power to run at full capacity, often due to thermal or power supply limitations.

**2. Feature Engineering:**

You already have the features that are useful for this task. Let's summarize the important ones:

* **Core Temperatures (avg) [°C]**: High temperatures can indicate thermal throttling.
* **CPU Package Power [W]**: Power limits can be related to power consumption.
* **Core Usage (avg) [%]**: High CPU usage combined with high temperatures can indicate throttling.
* **Core Clocks (avg) [MHz]**: Significant clock reductions can indicate throttling.

If needed, we can create more features to capture the relationships, such as:

* **CPU Frequency Change**: The rate at which the CPU clocks change can be an indicator of throttling.
* **Temperature Delta**: The change in temperature over time could also point to potential thermal throttling.

**3. Data Preprocessing:**

We’ll need to clean and preprocess the dataset:

* **Handle Missing Data**: Make sure there are no missing values in your features, or fill them appropriately.
* **Normalization**: For the Decision Tree model, you generally don’t need to normalize the data, as it handles varying scales well. However, ensuring features like temperature and power consumption are not on drastically different scales can help the model interpret the data better.
* **Encode Target Labels**: If you haven't labeled your data yet, you'll need to create a target variable:
  + 0 for "normal".
  + 1 for "thermal throttling".
  + 2 for "power limited".

You could label data by setting thresholds for temperature and power usage. For example:

* If Core Temperatures (avg) > 90°C, mark the entry as "thermal throttling."
* If CPU Package Power exceeds a certain threshold (like a predefined value), label it as "power limited."

**4. Splitting the Data:**

Split your data into training and testing datasets (typically 80% training and 20% testing).

from sklearn.model\_selection import train\_test\_split

# Assuming you have a DataFrame 'df' with your features and 'labels' as the target

X = df[['Core Clocks (avg) [MHz]', 'Ring/LLC Clock [MHz]', 'Core Usage (avg) [%]',

'Core Temperatures (avg) [°C]', 'CPU Package [°C]', 'CPU Package Power [W]']]

y = df['label'] # Labels for thermal throttling or power limited

# Split into training and test datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**5. Training a Decision Tree Classifier:**

Now, you can train a **Decision Tree** classifier. It will automatically learn to split the features to classify whether the CPU is thermal throttling or power limited.

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report, accuracy\_score

# Create the decision tree model

model = DecisionTreeClassifier(random\_state=42)

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model's performance

print(f'Accuracy: {accuracy\_score(y\_test, y\_pred)}')

print(f'Classification Report:\n{classification\_report(y\_test, y\_pred)}')

**6. Model Evaluation:**

Once you've trained the model, evaluate it using metrics like accuracy, precision, recall, and F1 score. You can also visualize the Decision Tree using the following:

from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

# Visualize the decision tree

plt.figure(figsize=(20,10))

plot\_tree(model, filled=True, feature\_names=X.columns, class\_names=['Normal', 'Thermal Throttling', 'Power Limited'])

plt.show()

**7. Model Optimization (Optional):**

To improve the performance of the model, you can:

* **Tune hyperparameters** using **GridSearchCV** or **RandomizedSearchCV** to find the optimal tree depth, minimum samples for splitting, etc.
* **Prune the tree** to avoid overfitting.
* **Use cross-validation** to ensure the model generalizes well to unseen data.

from sklearn.model\_selection import GridSearchCV

# Hyperparameter tuning with GridSearchCV

param\_grid = {

'max\_depth': [3, 5, 7, 10],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(DecisionTreeClassifier(random\_state=42), param\_grid, cv=5)

grid\_search.fit(X\_train, y\_train)

print("Best parameters found: ", grid\_search.best\_params\_)

**8. Deploying the Model:**

Once you have a trained model, you can integrate it into a pipeline where it processes logs, labels them, and makes predictions about thermal throttling or power limitations.

**Summary of Key Steps:**

1. **Label the data** based on temperature and power limits (define thresholds).
2. **Preprocess** the data (handle missing values, normalization if needed).
3. **Split the data** into training and testing sets.
4. **Train a Decision Tree** classifier on the labeled data.
5. **Evaluate** the model and tune hyperparameters if necessary.
6. **Deploy the model** for real-time log processing.

Let me know if you need further clarification or help with any of the steps!

**You said:**

how are we going to detect power limit throttling?

**ChatGPT said:**

Detecting **power limit throttling** can be a bit trickier since the signals aren’t as direct as thermal throttling. However, you can still make reasonable inferences from a few key features. Power limit throttling occurs when the CPU reaches a power consumption limit and is unable to perform at its full potential due to power constraints.

To detect power limit throttling, we can follow these steps:

**1. Define Power Limit Throttling:**

Power limit throttling typically occurs when the CPU is unable to increase its performance due to a restriction in available power. This can happen when:

* The **CPU Package Power** (W) reaches or exceeds a predefined power limit.
* **Core Clocks** (MHz) or **Core Usage** may be lower than expected for the workload due to power restrictions.

You can use thresholds to label your data as **power limited**. For example:

* If the **CPU Package Power** exceeds a certain threshold (e.g., 90W), we might assume the CPU is nearing or reaching its power limit.
* If the **Core Clocks** or **Core Usage** is unexpectedly low while the workload should demand more, that might also indicate power throttling.

**2. Features for Power Limit Detection:**

Key features that could indicate power throttling include:

* **CPU Package Power [W]**: This is the most direct indicator. If the CPU's power usage is consistently near the maximum allowed, it could suggest that power throttling is occurring.
* **Core Clocks [MHz]**: If the CPU is under power throttling, the **Core Clocks** could be lower than expected, especially during high load situations. If the CPU is limited in power but still under heavy load, the clocks may not increase, indicating power restrictions.
* **Core Usage [%]**: High usage combined with low clock speeds could indicate that the CPU is hitting power limits, as it may not be able to ramp up its frequency.
* **Temperature**: While **thermal throttling** would show a high temperature, **power throttling** might happen before the temperature even reaches high levels. However, if the CPU is running at a high temperature but the frequency doesn't increase (even under high load), power limitations could be at play.

**3. Labeling Power Limit Throttling:**

To label the data, we can combine the conditions:

* **Power limit** can be assumed if CPU Package Power exceeds a certain threshold (e.g., 80W, 90W).
* We can also check if the **Core Clocks** and **Core Usage** are low even though the CPU should ideally be running at higher speeds.

For example, you could define:

* If CPU Package Power > 90W and Core Clocks (avg) < X MHz (threshold based on typical usage), label it as **power limited**.
* If CPU Package Power is at or near a maximum and the **Core Clocks** are low, it's a strong indication of throttling.

**4. Feature Engineering for Power Throttling Detection:**

Some features might help refine detection:

* **Power to Frequency Ratio**: The ratio of **CPU Package Power** to **Core Clocks** could indicate how much power is being used relative to performance. If the ratio is high and the clocks are low, that could indicate power throttling.

Example feature:

 df['Power\_to\_Frequency\_Ratio'] = df['CPU Package Power [W]'] / df['Core Clocks (avg) [MHz]']

 **Delta of Power Consumption**: Calculate the rate of change in power usage. If the power usage is maxed out and can't increase, it might signal power throttling.

Example feature:

* df['Power\_Delta'] = df['CPU Package Power [W]'].diff() # Difference between consecutive readings

**5. Thresholds for Power Throttling:**

You can define a threshold to identify when the CPU is power-limited. Here are some example rules you could implement:

* **Power consumption limit**: Label a data point as **power limited** if the **CPU Package Power [W]** exceeds 90W or whatever the max threshold is for your system.
* **Low frequency despite high load**: If **Core Usage [%]** is high (e.g., > 80%) but **Core Clocks (avg)** are low, this could be an indicator of power limitation.

**6. Training the Model:**

You can now train the **Decision Tree** model to classify whether the CPU is **thermal throttling** or **power limited**.

1. Label your data:
   * If CPU Package Power > 90W and Core Clocks (avg) are low, label as **power limited**.
   * If Core Temperatures (avg) > 90°C, label as **thermal throttling**.
   * Otherwise, label as **normal**.
2. Preprocess and split the data:
   * Use features like CPU Package Power, Core Clocks, Core Usage, Temperature, etc.
   * Train the **Decision Tree** to predict whether the system is experiencing thermal throttling, power limitations, or normal operation.
3. The training would be similar to the previous code, but now your target variable (y) would have multiple classes (thermal throttling, power limited, normal).

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, accuracy\_score

# Labeling the data (example)

df['label'] = 'Normal'

df.loc[(df['Core Temperatures (avg) [°C]'] > 90), 'label'] = 'Thermal Throttling'

df.loc[(df['CPU Package Power [W]'] > 90) & (df['Core Clocks (avg) [MHz]'] < 1500), 'label'] = 'Power Limited'

# Preprocessing the features

X = df[['Core Clocks (avg) [MHz]', 'Ring/LLC Clock [MHz]', 'Core Usage (avg) [%]',

'Core Temperatures (avg) [°C]', 'CPU Package [°C]', 'CPU Package Power [W]']]

y = df['label']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train the decision tree model

model = DecisionTreeClassifier(random\_state=42)

model.fit(X\_train, y\_train)

# Evaluate the model

y\_pred = model.predict(X\_test)

print(f'Accuracy: {accuracy\_score(y\_test, y\_pred)}')

print(f'Classification Report:\n{classification\_report(y\_test, y\_pred)}')

**7. Model Evaluation and Tuning:**

Finally, evaluate your model’s accuracy and performance. You can tune the hyperparameters of the decision tree (like **max\_depth**, **min\_samples\_split**) using **GridSearchCV** or **RandomizedSearchCV** for better results.

**Summary:**

* **Power limit throttling** occurs when the CPU reaches its power consumption limit.
* We can label power limit throttling based on thresholds for **CPU Package Power** and **Core Clocks**.
* Features like **Power-to-Frequency Ratio**, **Core Usage**, and **Power Delta** can be useful for detecting throttling.
* Train a **Decision Tree** model to classify CPU states into **thermal throttling**, **power limited**, and **normal** states.

Let me know if you need further help or details!

**You said:**

do you know about the parameters provided by hwinfo?

**ChatGPT said:**

Yes, I'm familiar with **HWiNFO**—a popular system monitoring tool that provides detailed information about various hardware components, including CPU, GPU, memory, and more. HWiNFO outputs a wide variety of parameters, and from what you've shared, it looks like you're using a subset related to **CPU monitoring**.

Here are some common **CPU-related parameters** typically provided by **HWiNFO** that could be relevant to your analysis:

**Common CPU Parameters from HWiNFO:**

1. **Core Clocks (avg) [MHz]**:
   * Represents the average clock speed (frequency) of the CPU cores over a given period. This is crucial for understanding CPU performance. Lower clock speeds could indicate throttling, either thermal or power-related.
2. **Ring/LLC Clock [MHz]**:
   * Refers to the clock speed of the **Ring Bus** or **Last-Level Cache (LLC)**. The Ring Bus connects various parts of the CPU, such as cores, caches, and the memory controller. Lower speeds here could also indicate a performance bottleneck.
3. **Core Usage (avg) [%]**:
   * This indicates the average percentage of usage across all CPU cores over a given period. High CPU usage with low clock speeds could indicate thermal or power throttling.
4. **Core Temperatures (avg) [°C]**:
   * The average temperature of the CPU cores. High temperatures are often the main indicator of **thermal throttling**. When temperatures exceed safe thresholds, the CPU will throttle itself to prevent overheating.
5. **CPU Package [°C]**:
   * The overall temperature of the CPU package, which includes all cores, caches, and other related components. This can be used as an additional indicator of whether the CPU is overheating.
6. **CPU Package Power [W]**:
   * The total power consumption of the CPU package. If this exceeds certain limits, it might indicate **power-limited throttling**. This value, when combined with clock speeds and temperature, can help identify whether the CPU is being limited by power delivery.

**Other Useful Parameters (Additional Context):**

While not necessarily directly relevant to thermal or power throttling, here are some additional metrics HWiNFO may provide:

1. **Core Voltage [V]**:
   * The voltage supplied to the cores. Lower or fluctuating voltages could indicate power issues, and the voltage may change depending on throttling or load.
2. **Package Power Limit (W)**:
   * This represents the **power limit** defined by the CPU manufacturer. If the **CPU Package Power** exceeds this value, it may indicate **power throttling**.
3. **TDP (Thermal Design Power)**:
   * This is the amount of heat the CPU is designed to dissipate under normal operation. It can serve as a baseline to compare against actual power consumption to detect throttling.
4. **Fan Speed [RPM]**:

* The speed of the cooling fan(s). A sudden increase in fan speed combined with high temperatures could indicate thermal throttling, as the system tries to cool down the CPU.

1. **C-State [#]**:

* This is related to the CPU’s **idle states**. Higher C-states (C1, C2, etc.) represent deeper idle states. If the system is in a deeper C-state under load, it might indicate that it is being limited by power or thermal conditions.

1. **P-State [#]**:

* **Performance states** (P-states) refer to the CPU’s operating frequency and voltage. A change to a low P-state under high load could suggest throttling due to either power or temperature constraints.

**Power Throttling Specifics:**

To detect **power-limited throttling**, the most relevant parameters would likely be:

* **CPU Package Power [W]**: Monitor if it exceeds typical limits.
* **Core Clocks (avg) [MHz]**: Low clocks could indicate throttling.
* **CPU Package Power Limit (W)**: Check if the CPU is hitting its power limit.
* **Core Usage (avg) [%]**: If the CPU usage is high but clocks are low, it's a sign of potential throttling.

**Thermal Throttling Specifics:**

For **thermal throttling**, the most relevant parameters would include:

* **Core Temperatures (avg) [°C]**: High temperatures (e.g., >90°C) are a strong indicator of thermal throttling.
* **CPU Package [°C]**: If this temperature is also high, it supports the likelihood of thermal throttling.

**Example of Using HWiNFO Data:**

For detecting **thermal throttling** or **power-limited throttling**, you might define rules based on these thresholds, such as:

* **Thermal Throttling**: If Core Temperatures (avg) > 90°C or CPU Package [°C] > 90°C.
* **Power Limited**: If CPU Package Power [W] > 90W (based on the system's maximum power delivery capacity) and Core Clocks (avg) < 1500 MHz.

These can be used as labels to train a model to classify CPU states as either **thermal throttling**, **power-limited**, or **normal**.

**Conclusion:**

By combining the information provided by HWiNFO, such as temperature, power usage, and clock speeds, you can easily identify patterns indicative of thermal throttling or power limitations. For example, high **core temperatures** with reduced **clock speeds** would point toward thermal throttling, while high **CPU package power** without corresponding high clock speeds could signal power throttling.

Would you like to proceed with more detailed steps on how to preprocess and label this data for training, or do you need more help with HWiNFO parameters?

**You said:**

yea can you provide me with a list of all those features that can help me define the feature set for my data on which i can train my model? currently these are features that I am working with: core clocks cpu package temp core temp package power i have removed date and time since they are not very useful for model.

**ChatGPT said:**

Sure! You’ve already selected some of the most important features for detecting **thermal throttling** and **power-limited throttling**. However, there are a few additional features from HWiNFO that you can include to improve the model’s ability to detect throttling, based on how well they capture CPU behavior under different conditions.

Here’s a more comprehensive list of features that can help in defining your feature set:

**1. Core Clocks (avg) [MHz]:**

* This is one of the most important features. **Lower core clock speeds** are often associated with throttling, especially if the CPU should be running at higher speeds during a load.

**2. Core Temperatures (avg) [°C]:**

* **High core temperatures** can be an indicator of **thermal throttling**. When the CPU gets too hot, the system may reduce performance to prevent overheating.

**3. CPU Package Temperature [°C]:**

* Similar to core temperature, this reflects the overall temperature of the CPU package. A high temperature often correlates with thermal throttling.

**4. CPU Package Power [W]:**

* **Power consumption** is an essential factor. If the CPU is consuming too much power, it may be **power-limited** or could be throttling due to reaching its power limits.

**5. Core Usage (avg) [%]:**

* This is a crucial feature to track. **High core usage** (over 80-90%) along with **low core clock speeds** can indicate throttling (either thermal or power-related). If the CPU usage is high but the clocks are low, it might mean the CPU is throttling due to power or thermal constraints.

**6. Ring/LLC Clock [MHz]:**

* The **Ring Bus** or **Last-Level Cache (LLC)** clock speeds could impact performance. **Lower clock speeds** here might indicate limitations in CPU performance due to power or thermal factors, as these components are involved in data transfer between cores and other CPU components.

**7. CPU Frequency [MHz] (if available):**

* Some systems report the actual CPU frequency, which can be a more direct measure of throttling. If the frequency is lower than expected, it could be due to power or thermal limits.

**8. CPU Package Power Limit [W]:**

* This is the **maximum power** the CPU is designed to draw. When the **CPU Package Power** exceeds this value, it indicates the system is likely **power-limited**.

**9. Core Voltage [V] (if available):**

* The voltage supplied to the CPU cores. A significant **drop in voltage** can sometimes be an indicator of power throttling, especially if the CPU is trying to prevent overheating or reduce power consumption.

**10. Fan Speed [RPM] (if available):**

* **Fan speed** is an indirect indicator. If the fan speed increases suddenly, it may be a reaction to increased temperatures, which is often seen during **thermal throttling**.

**11. C-State [#]:**

* This is related to the CPU's **idle states**. A deep **C-State** under load could indicate the CPU is throttling due to either power limitations or temperature constraints. This is often seen in energy-saving modes, but it could also appear if power delivery is insufficient.

**12. P-State [#]:**

* **P-States** refer to the CPU's operating performance state. If the CPU is running at a low P-state (e.g., lower clock speed and voltage), it can indicate throttling due to power or thermal constraints.

**13. Package Cores Active:**

* The **number of active cores** could be an indicator. A CPU with fewer active cores under heavy load might be throttling, either to avoid overheating or due to power limitations.

**14. Temperature Delta (Change in temperature over time):**

* Calculating the **delta** in temperature over a given period could give insights into **thermal throttling**. Rapid temperature increases could indicate a performance hit due to heating.

**15. Power to Frequency Ratio:**

* **Power consumption per MHz** can indicate how efficiently the CPU is running. If the ratio is high, it could suggest that the CPU is using too much power for a given clock speed, possibly signaling power throttling.

Example:

df['Power\_to\_Frequency\_Ratio'] = df['CPU Package Power [W]'] / df['Core Clocks (avg) [MHz]']

**16. Power Delta (Change in power consumption over time):**

* Tracking how **CPU power consumption** changes over time can help detect **power limit throttling**. If power consumption is stuck at or near its maximum value, it could indicate that power throttling is occurring.

Example:

df['Power\_Delta'] = df['CPU Package Power [W]'].diff() # Difference between consecutive readings

**Suggested Feature Set for Your Model:**

To create a more complete feature set for training your **Decision Tree** model to detect **thermal throttling** and **power-limited throttling**, you can include the following features:

1. **Core Clocks (avg) [MHz]**
2. **Core Temperatures (avg) [°C]**
3. **CPU Package Temperature [°C]**
4. **CPU Package Power [W]**
5. **Core Usage (avg) [%]**
6. **Ring/LLC Clock [MHz]**
7. **CPU Package Power Limit [W]** (if available)
8. **Core Voltage [V]** (if available)
9. **Fan Speed [RPM]** (if available)
10. **C-State [#]** (if available)
11. **P-State [#]** (if available)
12. **Temperature Delta** (change in temperature over time)
13. **Power to Frequency Ratio**
14. **Power Delta** (change in power consumption over time)

By including a diverse set of features, you enable the model to better understand various aspects of CPU behavior, whether it’s due to **thermal throttling**, **power throttling**, or normal operation. This should help the model perform well in identifying patterns and making accurate predictions.

**Next Steps:**

1. **Data Preprocessing**:
   * Clean the data and handle missing values.
   * Normalize or scale the data if required (though Decision Trees usually don’t need this).
   * Label your data (thermal throttling, power-limited throttling, normal).
2. **Train the Model**:
   * Use the features above to train a **Decision Tree Classifier**.
3. **Model Evaluation**:
   * Evaluate the model using metrics like accuracy, precision, recall, and F1 score.
   * Tune hyperparameters to improve performance.

Let me know if you need help with any part of the process, such as data labeling, model training, or feature engineering!

Top of Form

Bottom of Form

ChatGPT can make mistakes. Check important info.