**Experiment Summary:**

The model used was Gemini 2.5 Flash with the Google GenAI library. Different parameters were tested such as temperature, top p, top k, and max output tokens.

**Observations:**

**Configs for Logic:**

* **Temperature = 0.3**
* **Max Output Tokens: 1000**
* **Top P: 0.8**
* **Top K: 64**

**Default configs are used for the other prompts as it was the most effective.**

**Logic:** What are the number of "r" in the word strawberry?

There are \*\*3\*\* "r"s in the word "strawberry".

**Instructional:** Explain how photosynthesis works to a 10-year-old using simple language.

Hey there! Have you ever wondered how plants eat, especially since they don't have mouths or go to the grocery store? Well, they have a super cool superpower called \*\*photosynthesis\*\*!

Think of a plant like a little chef, and its leaves are its tiny kitchens. Photosynthesis is how plants cook their own food.

Here's how it works, step-by-step:

1. \*\*Ingredients, Please!\*\*

\* \*\*Sunlight (Energy):\*\* This is like the oven or stove that gives the plant energy to cook. Plants have a special green stuff in their leaves called \*\*chlorophyll\*\* (that's why most leaves are green!). Chlorophyll's job is to trap sunlight, just like a solar panel.

\* \*\*Water (Liquid):\*\* Plants suck up water from the soil through their roots, like drinking through a straw. This water travels up to the leaves.

\* \*\*Carbon Dioxide (Air):\*\* This is an invisible gas that's all around us in the air (it's actually what we breathe

\*out\*!). Plants "breathe" in carbon dioxide through tiny little holes on their leaves, like microscopic mouths.

2. \*\*The Cooking Begins!\*\*

\* Inside the leaf-kitchens, the sunlight gives the chlorophyll the energy it needs.

\* The chlorophyll then takes the water and carbon dioxide and, using all that sunlight energy, mixes them up and magically changes them into something new!

3. \*\*The Delicious Meal!\*\*

\* \*\*Sugar (Food!):\*\* The main thing plants make is a type of sugar. This sugar is the plant's food! It gives the plant all the energy it needs to grow tall, make new leaves, flowers, and even yummy fruits or vegetables for us to eat!

\* \*\*Oxygen (Extra Gas):\*\* As a sort of "waste product" or "leftover," plants also release oxygen into the air. This

is the air that \*we\* and animals breathe to live! So, while plants are busy making their food, they're also making the air we need to survive. How cool is that?

\*\*So, in a nutshell:\*\*

Photosynthesis is when plants use \*\*sunlight\*\*, \*\*water\*\*, and \*\*carbon dioxide\*\* to make their own \*\*sugar (food)\*\* and \*\*oxygen\*\*. It's how they grow, and it's super important for all life on Earth! Without plants and photosynthesis, we wouldn't have clean air to breathe or much food to eat. Plants are truly amazing chefs!

**Code Generation:** Write a Python function that takes a list of numbers and returns the median. Include comments.

Here's a Python function to calculate the median of a list of numbers, complete with comments and a docstring.

python

def calculate\_median(numbers\_list):

"""

Calculates the median of a list of numbers.

The median is the middle value in a sorted list of numbers.

If the list has an odd number of elements, the median is the middle element.

If the list has an even number of elements, the median is the average of

the two middle elements.

Args:

numbers\_list (list): A list of numerical values (integers or floats).

Returns:

float: The median of the numbers in the list.

Raises:

ValueError: If the input list is empty.

"""

# Step 1: Handle the edge case of an empty list.

# The median is undefined for an empty set of numbers.

if not numbers\_list:

raise ValueError("Input list cannot be empty. Median is undefined for an empty list.")

# Step 2: Sort the list in ascending order.

# This is a crucial step because the median is based on the ordered data.

sorted\_list = sorted(numbers\_list)

# Step 3: Get the number of elements in the sorted list.

n = len(sorted\_list)

# Step 4: Determine if the number of elements is odd or even.

if n % 2 == 1:

# If n is odd, the median is the middle element.

# Integer division (//) correctly gives the middle index.

# For example, if n=5, n//2 = 2, so sorted\_list[2] is the middle element.

median = float(sorted\_list[n // 2])

else:

# If n is even, the median is the average of the two middle elements.

# For example, if n=4, n//2 = 2.

# The two middle elements are at indices (n//2 - 1) and (n//2).

# So, for n=4, they are at indices 1 and 2.

mid1 = sorted\_list[n // 2 - 1]

mid2 = sorted\_list[n // 2]

median = (mid1 + mid2) / 2.0 # Use 2.0 to ensure float division

return median

# --- Example Usage ---

# Test cases for odd number of elements

list1 = [1, 2, 3, 4, 5]

print(f"List: {list1}, Median: {calculate\_median(list1)}") # Expected: 3.0

list2 = [10, 5, 20, 15, 0]

print(f"List: {list2}, Median: {calculate\_median(list2)}") # Expected: 10.0 (sorted: [0, 5, 10, 15, 20])

list3 = [7.5]

print(f"List: {list3}, Median: {calculate\_median(list3)}") # Expected: 7.5

# Test cases for even number of elements

list4 = [1, 2, 3, 4]

print(f"List: {list4}, Median: {calculate\_median(list4)}") # Expected: 2.5 (average of 2 and 3)

list5 = [50, 10, 30, 20]

print(f"List: {list5}, Median: {calculate\_median(list5)}") # Expected: 25.0 (sorted: [10, 20, 30, 50], average of 20 and 30)

list6 = [3.1, 1.1, 4.1, 2.1]

print(f"List: {list6}, Median: {calculate\_median(list6)}") # Expected: 2.6 (sorted: [1.1, 2.1, 3.1, 4.1], average of 2.1 and 3.1)

# Test case for an empty list (should raise an error)

try:

empty\_list = []

print(f"List: {empty\_list}, Median: {calculate\_median(empty\_list)}")

except ValueError as e:

print(f"Error for empty list: {e}")

**Comparative Analysis:** Compare an apple and orange. List 3 key differences.

While both are popular, nutritious fruits, apples and oranges have distinct characteristics that set them apart.

Here are 3 key differences:

1. \*\*Skin/Peel:\*\*

\* \*\*Apple:\*\* Has a thin, smooth, and edible skin that is usually red, green, or yellow.

\* \*\*Orange:\*\* Has a thicker, dimpled, and inedible peel that is removed before eating (and is typically orange in color).

2. \*\*Internal Structure:\*\*

\* \*\*Apple:\*\* Consists of solid, firm flesh surrounding a central core containing small seeds.

\* \*\*Orange:\*\* Is composed of distinct, easily separable segments filled with juicy pulp vesicles, which may or may not contain seeds.

3. \*\*Flavor and Texture Profile:\*\*

\* \*\*Apple:\*\* Generally offers a crisp, crunchy texture with a balanced sweet-tart flavor.

\* \*\*Orange:\*\* Is known for its very juicy, pulpy texture and a distinctive tangy, citrusy, and often sweeter taste.

**Creative Writing:** Write a very short story about a boy meeting his long-lost brother.

Leo stood by the platform clock, heart thrumming. In his hand, a worn photo – a blurry child’s face, years younger, yet still unmistakably like his own.

A train hissed to a halt. Passengers poured out, a river of faces. Then he saw him. A tall young man, maybe ten years older, with the same unruly brown hair and – unmistakable – his own green eyes staring back.

Leo’s breath caught. The man hesitated, then a slow, unsure smile spread across his face, mirroring Leo’s own.

"Leo?" the man’s voice was rough, almost a whisper.

"Alex?" Leo managed, the name feeling foreign yet utterly right.

Time seemed to dissolve. The next second, Leo was barreling forward, not caring who watched. He crashed into the man, who

wrapped strong arms around him. It wasn't just a hug; it was years of missing pieces clicking into place.

**Academic Summary:** Summarize the key contributions of Einstein's theory of relativity in short bullet points.

Einstein's theory of relativity (encompassing both Special and General Relativity) revolutionized our understanding of space, time, gravity, and the universe. Its key contributions include:

\* \*\*Constancy of Light Speed:\*\* Established that the speed of light in a vacuum is constant for all observers, regardless of their relative motion.

\* \*\*Spacetime Unification:\*\* Unified space and time into a single, four-dimensional "spacetime" continuum, where events

are relative to an observer's motion.

\* \*\*Relativistic Effects:\*\* Introduced concepts like \*\*time dilation\*\* (time passing slower for objects in motion relative to a stationary observer) and \*\*length contraction\*\* (objects appearing shorter in the direction of motion).

\* \*\*Mass-Energy Equivalence (E=mc²):\*\* Postulated that mass and energy are interchangeable and equivalent, forming the basis for nuclear energy and weapons.

\* \*\*Gravity as Spacetime Curvature:\*\* Redefined gravity not as a force, but as the curvature of spacetime caused by the

presence of mass and energy.

\* \*\*Predictive Power:\*\* Successfully predicted and explained phenomena such as:

\* The \*\*bending of light by massive objects\*\* (gravitational lensing).

\* The \*\*anomalous precession of Mercury's orbit\*\*.

\* \*\*Gravitational redshift\*\* (light losing energy as it escapes a gravitational field).

\* \*\*Prediction of Gravitational Waves:\*\* Predicted the existence of ripples in spacetime caused by accelerating masses,

later directly detected.

\* \*\*Prediction of Black Holes:\*\* Implied the existence of regions where gravity is so intense that nothing, not even light, can escape.

**Reflection:**

The model performs well in general through all types of prompts. This may be due to how polished the Gemini 2.5 Flash model is, performing well in logic questions such as the infamous strawberry problem that stumped ChatGPT in the past. The only improvement that could be applied is the generation of more prompts that the model could use using the logic configurations of temperature and top p resulted in no results for some prompts. Due to this limitation, output length limitations were used in the prompts instead.