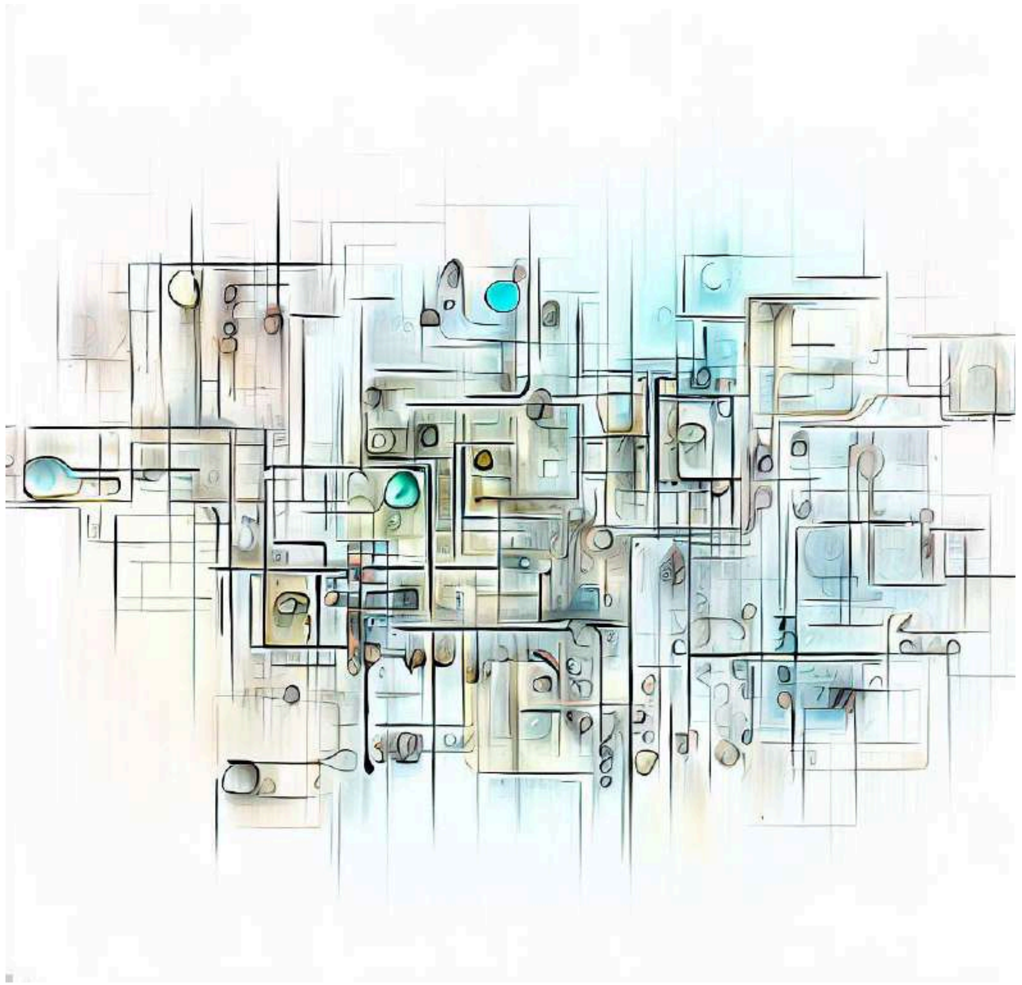
# Generative AI with Google

# Getting Started with Generative AI

- Artificial Intelligence is in its **second golden phase** (IMHO):
  - **First Phase** - Early successes in the 1960s and 1970s
  - **Second Phase** - Last two decades or so driven by:
    - New algorithms
    - Massive datasets
    - Powerful computing hardware

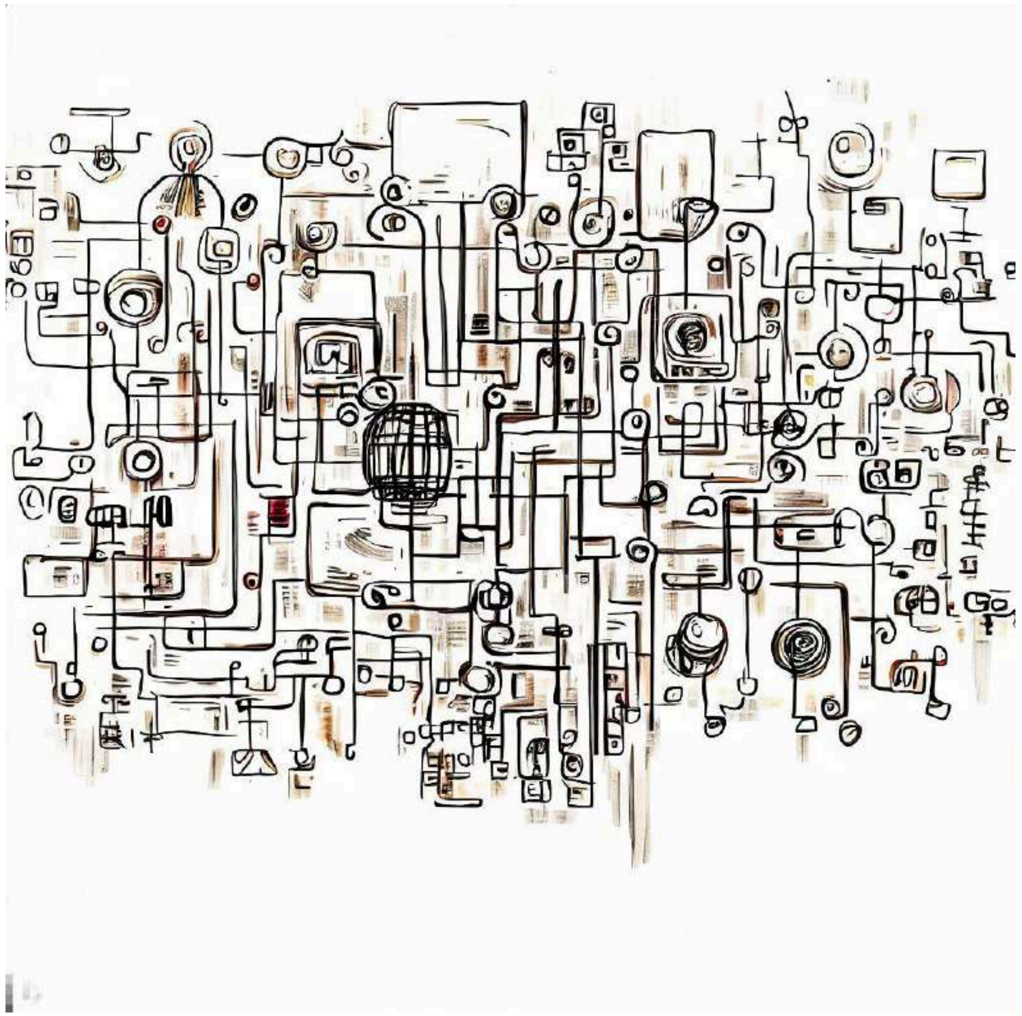
## Getting Started with Generative AI - 2

- HOWEVER, **building AI solutions has NOT been easy:**
  - Scarcity of skills
  - Need to create/manage massive datasets
  - Complex infrastructure needs
  - Complexity of managing AI models



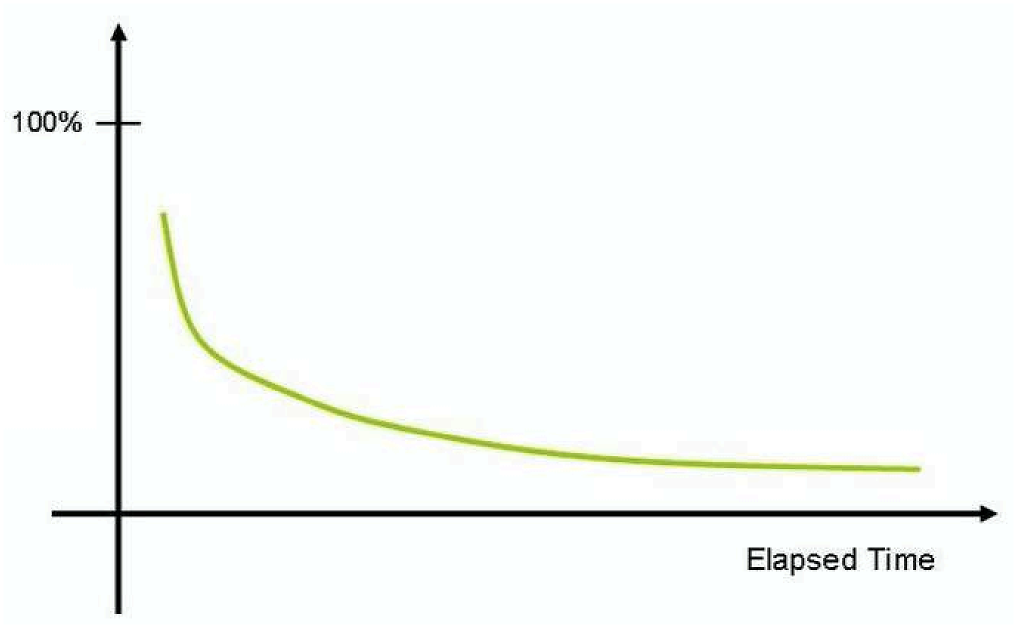
## Getting Started with Generative AI - 3

- How can you **use AI WITHOUT AI skills**/complex infrastructure?
  - Use low-code AI platforms (AutoML)
  - Use pre-trained models (especially Generative AI models)
  - Google offers a plethora of AI solutions **around Generative AI**:
    - **Gemini**: Chatbot
    - **PaLM API & MakerSuite**: Easy to consume APIs
    - **Gen. AI Studio (Vertex AI)**: Google Cloud Service
- **Our Goal** : Help you understand AI, ML and Generative AI while exploring Generative AI solutions offered by Google



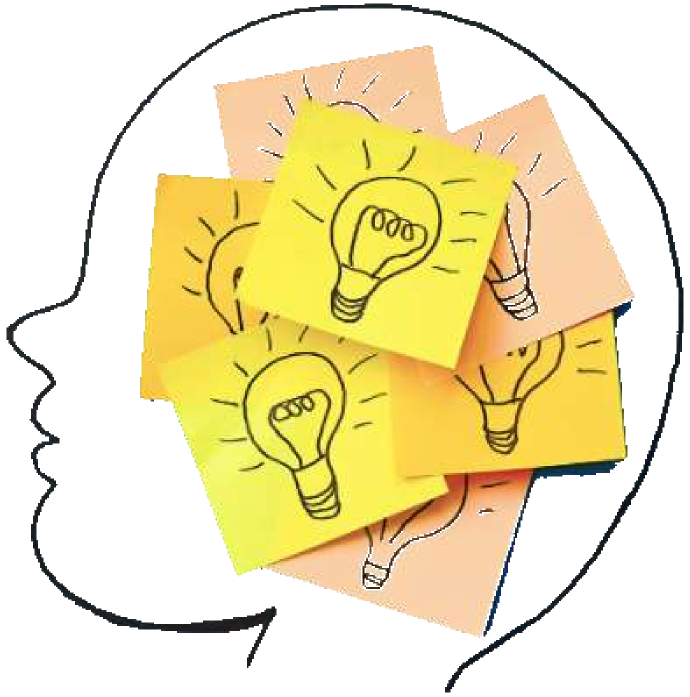
## How do you put your best foot forward?

- **Learning Gen. AI can be tricky:**
  - Lots of new terminology
  - Lots of services
- As time passes, we forget things
- How do you **improve your chances** of remembering things?
  - **Active learning** - think and make notes
  - **Review the presentation** once in a while



## Our Approach

- Three-pronged approach to reinforce concepts:
  - Presentations (Video)
  - Demos (Video)
  - **Two kinds of quizzes**:
    - Text quizzes
    - Video quizzes
- (Recommended) Take your time. Do not hesitate to replay videos!
- (Recommended) Have Fun!



## FASTEST ROADMAPS

in28minutes.com

In28 Minutes

Google Cloud Certifications

Azure Certifications

AWS Certifications

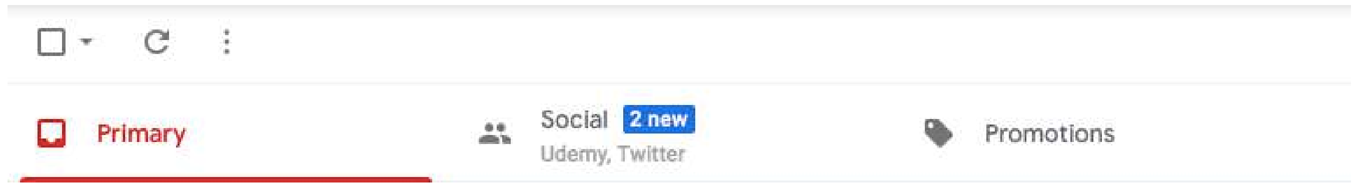
DevOps

Java Full Stack

Java Microservices

## Artificial Intelligence

# Artificial Intelligence - All around you



- Self-driving cars
- Spam Filters
- Email Classification
- Fraud Detection

## What is AI? (Oxford Dictionary)

In28 Minutes

*The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages*

## What is AI? (in Simple Terms)

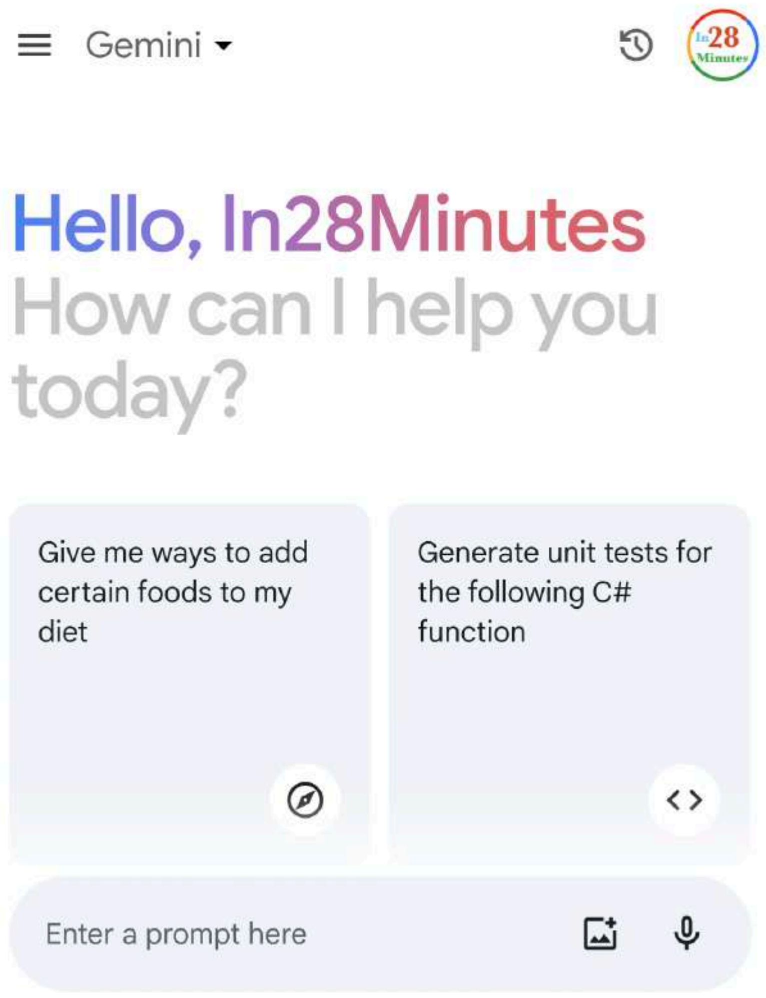
- **Goal of AI**: Create machines that can simulate human-like intelligence and behavior
  - Play Chess
  - Play Go
  - Make purchase decisions
  - Drive a car
  - Write an essay!

## Understanding Types of AI

- **Strong artificial intelligence (or general AI):** Intelligence of machine = Intelligence of human
  - A machine that can solve problems, learn, and plan for the future
  - An expert at everything
    - Including learning to play all sports and games!
  - Learns like a child, building on it's own experiences
  - We are far away from achieving this!
    - Estimates: few decades to never
- **Narrow AI (or weak AI):** Focuses on specific task
  - Example: Self-driving car
  - Example: Playing Chess
  - Example: Predicting House Price

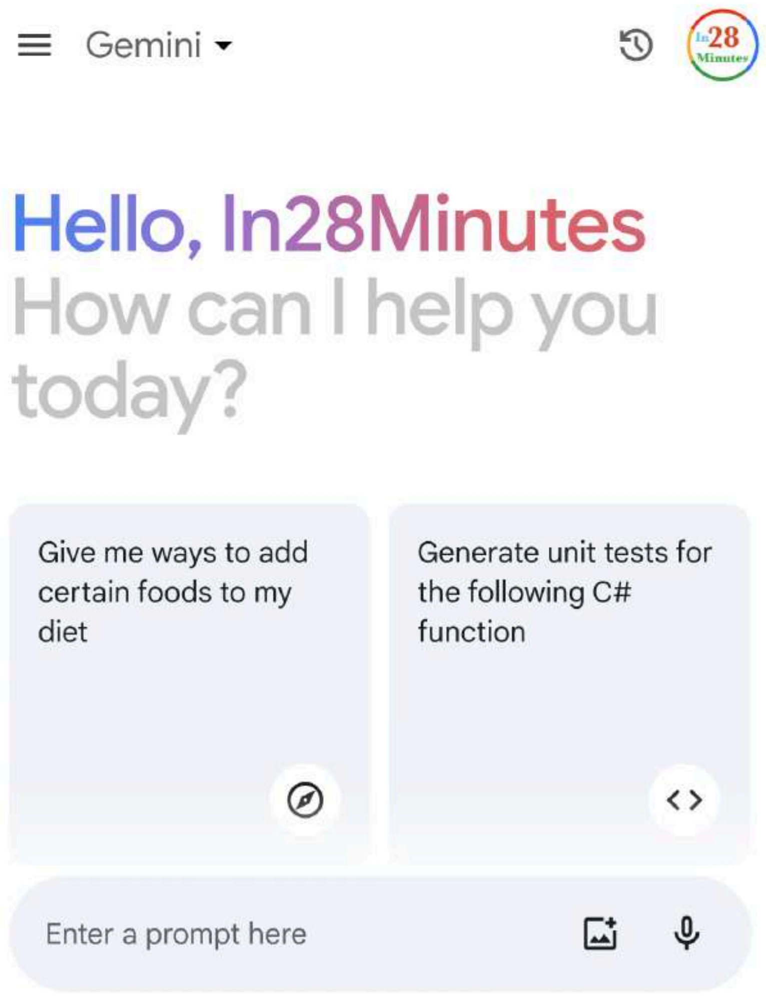
## Playing with Gemini

- **Gemini**: Google's Generative AI Chatbot!
  - I'm Gemini, your creative and helpful collaborator. I have limitations and won't always get it right, but your feedback will help me improve.
- **A Demo of Gemini** (earlier called Bard):
  - You are Lex Friedman. You are going to interview Sachin Tendulkar tomorrow. What are the FIRST FIVE questions that you are going to ask?
  - Act as Sachin Tendulkar. You are meeting Roger Federer. What questions would you ask?
  - Generate a bulleted list of items I need for an 15 day Everest Base Camp trek
    - I will be staying in tea houses on the trek. Can you update the list?


Gemini
Hello, In28Minutes
How can I help you today?
Give me ways to add certain foods to my diet
Generate unit tests for the following C# function
Enter a prompt here

## Playing with Gemini - Coding, Learning and Design

- Write a Python function to determine if a year is a leap year
- I'm learning Python for loop. Give me a few exercises to try?
- Can you design a REST API for todos? Give me an example request and response for each.
- I want to store information about courses, students, enrollments and reviews in a relational table. Can you suggest a structure?
- I like learning concepts using a lot of examples. What would be the books you would recommend to learn Design Patterns?



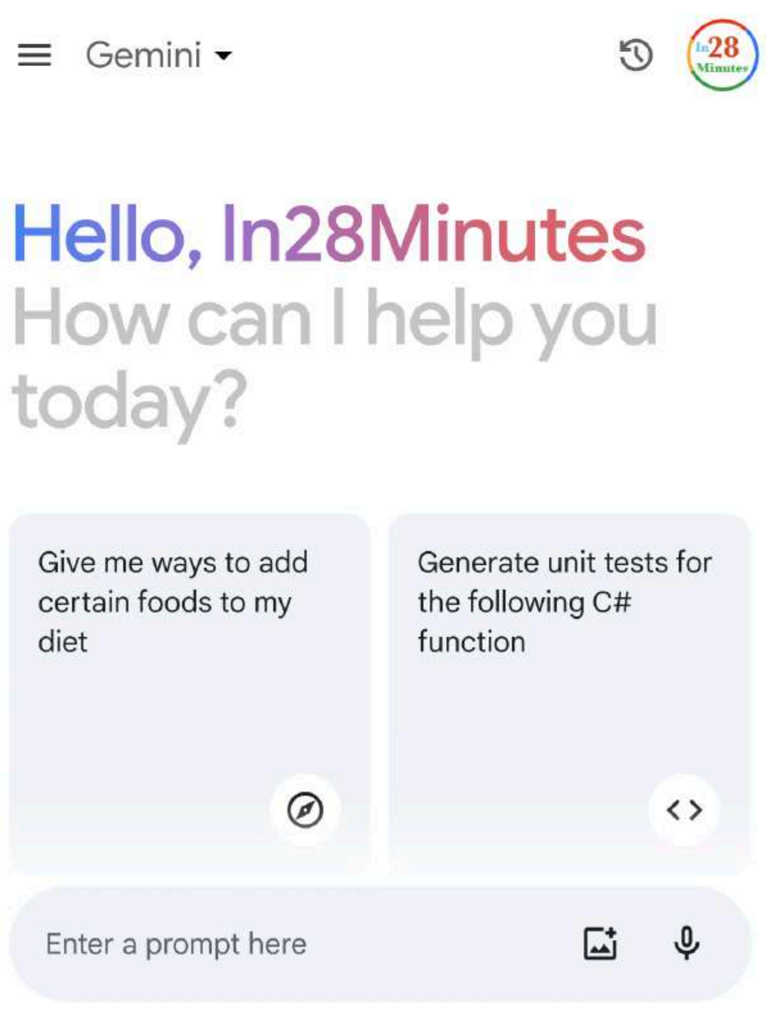
Hello, In28Minutes
How can I help you today?

Give me ways to add certain foods to my diet

Generate unit tests for the following C# function

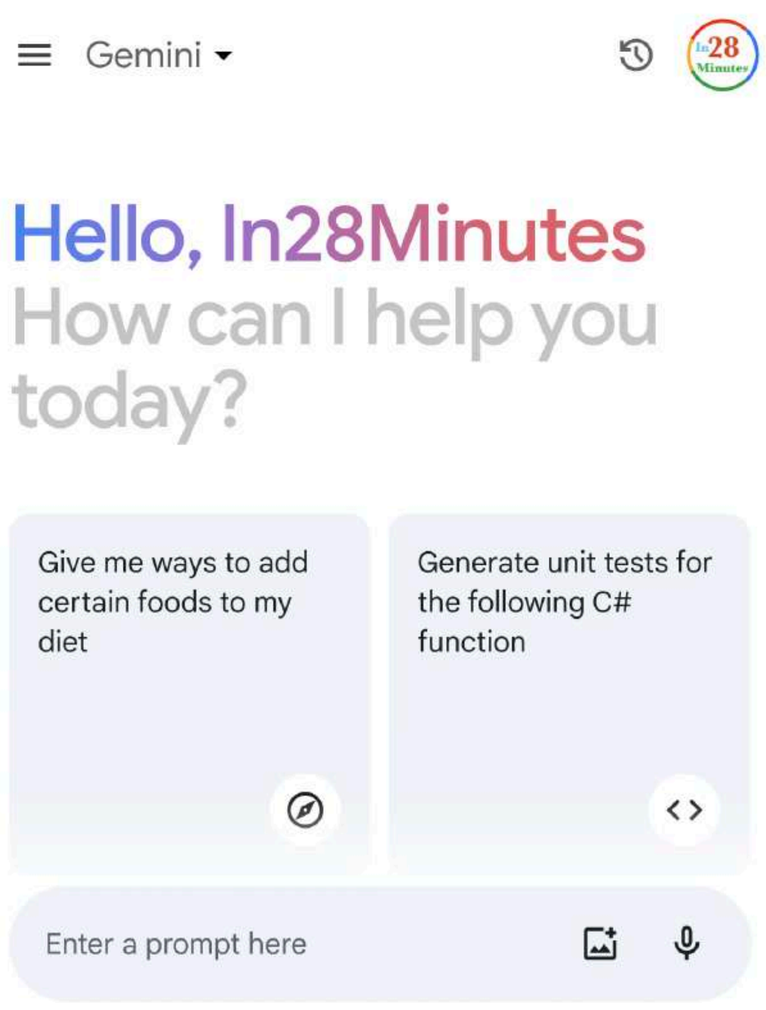
## Playing with Gemini - Exploring Technology

- Can you make list of Top 10 technologies that I might want to learn as a cloud engineer?
- For a new project, I'm considering React and Angular as front end frameworks.
  - Can you compare them and present the results in a tabular format? Feature/Factor in the column and the framework in the row.
- I like to learn in a step-by-step approach by breaking down complex concepts into smaller, more manageable parts.
  - How can I learn Docker? Give me a list of 10 step by step exercises I can begin with. Make sure you order them in decreasing order of difficulty. Present the results in a table.


Gemini
Hello, In28Minutes
How can I help you today?
Give me ways to add certain foods to my diet
Generate unit tests for the following C# function
Enter a prompt here

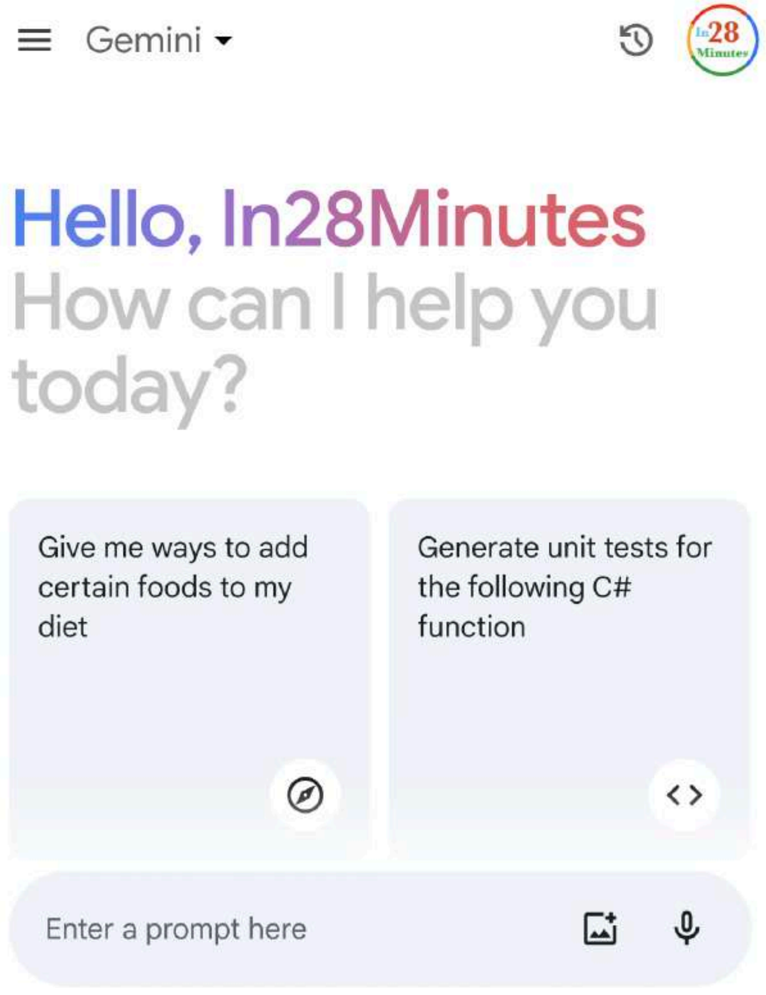
# Playing with Gemini - Generating Ideas

- "Playing with Gemini - Generating Ideas". I want to create a few prompts under this heading exploring different DevOps Tools. Can you suggest a few prompts?
- How can I make it easy for a beginner to learn a new programming language / tool / technology / framework / methodology? One example is to use analogies. Can you think about other similar things I can do. Make sure you order them in decreasing order of importance. Rate importance from 1-100. 1 being least important. Present the results in a table


Gemini
Hello, In28Minutes
How can I help you today?
Give me ways to add certain foods to my diet
Generate unit tests for the following C# function
Enter a prompt here

## Playing with Gemini - Observations

- Gemini MAY display inaccurate or offensive information
- BUT it does provide a lot of value if you understand its limitations and know when/how to use it
- How does Gemini work?
  - Artificial Intelligence
  - Machine Learning
  - Generative AI
  - Large Language Models
  - Foundation Models



Gemini

Hello, In28Minutes
How can I help you today?

Give me ways to add certain foods to my diet

Generate unit tests for the following C# function

Enter a prompt here

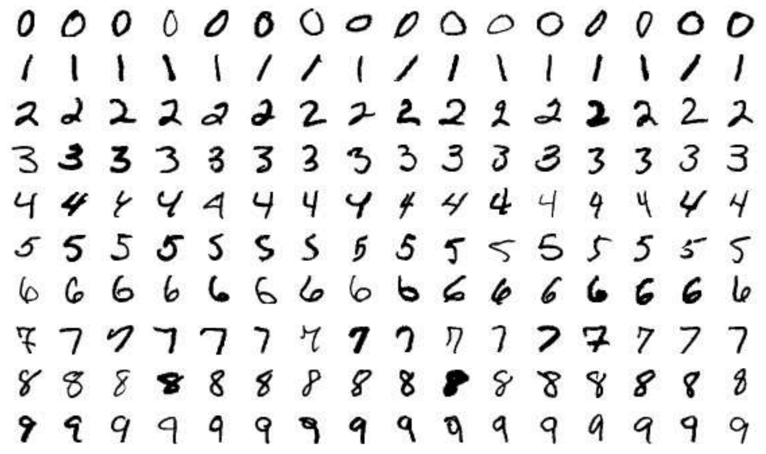
## AI vs ML vs Generative AI

- **Goal of AI**: Create machines that can simulate human-like intelligence and behavior
  - What is ML?
  - How does Generative AI fit in?
- Let's get started on a Journey!

## Machine Learning vs Traditional Programming

- **Traditional Programming:** Based on Rules
  - IF this DO that
  - Example: Predict price of a home
    - Design an algorithm taking all factors into consideration:
      - Location, Home size, Age, Condition, Market, Economy etc
- **Machine Learning:** Learning from Examples (NOT Rules)
  - Give millions of examples
  - Create a Model
  - Use the model to make predictions!

| Home size (Square Yds) | Age | Condition (1-10) | Price $$$ |
|---|---|---|---|
| 300 | 10 | 5 | XYZ |
| 200 | 15 | 9 | ABC |
| 250 | 1 | 10 | DEF |
| 150 | 2 | 34 | GHI |

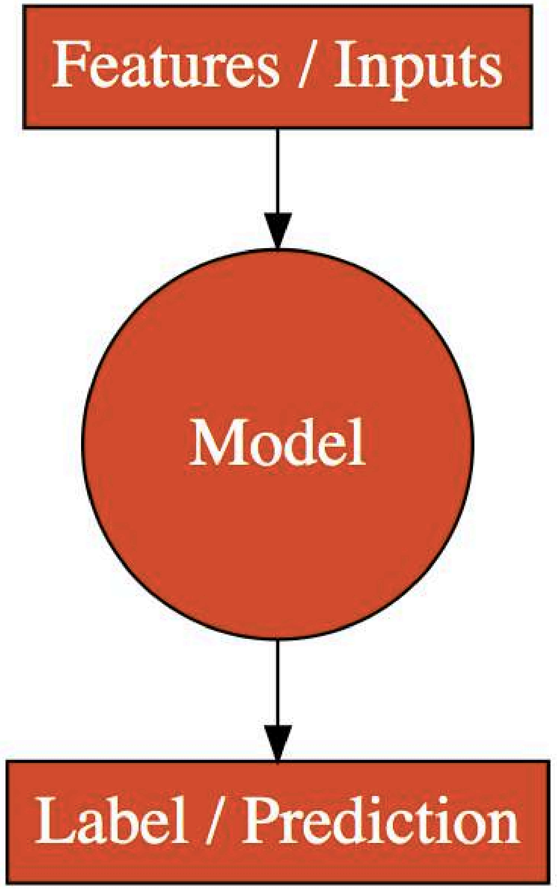


# Machine Learning Fundamentals - Scenarios

| Scenario | Solution |
|---|---|
| Categorize: Building a computer system as intelligent as a human. An expert at everything (all sports and games!) | Strong AI |
| Categorize: Building a computer system that focuses on specific task (Self-driving cars, virtual assistants, object detection from images) | Narrow AI (or weak AI) |
| Category of AI that focuses on learning from data (examples) | Machine learning |
| How is ML different from traditional programming? | Traditional Programming: Rules. Machine Learning: Examples |

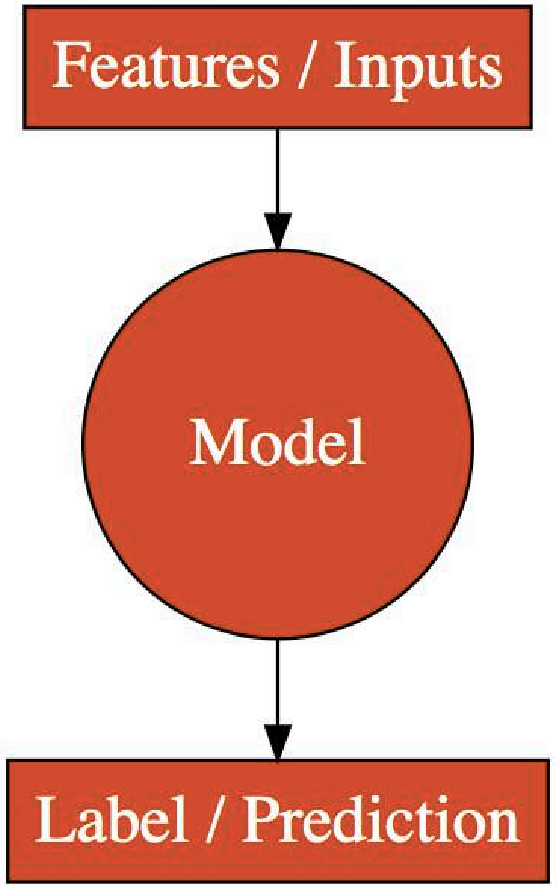
## Machine Learning - Making Prediction

- **Goal**: Make a Good Prediction
  - Give inputs to a model
  - Model returns the prediction
  - Inputs are called **Features**
  - Prediction is called **Label**
  - **Example**: House Price Prediction Model
    - **Label**: price
    - **Features**:
      - area: Total area of house (m^2)
      - rooms: No. of rooms
      - bedrooms: No. of bedrooms
      - furniture: Is it furnished?
      - floor: Which floor?
      - age: How many years?
      - balcony: has balcony or not
      - garden: has garden or not



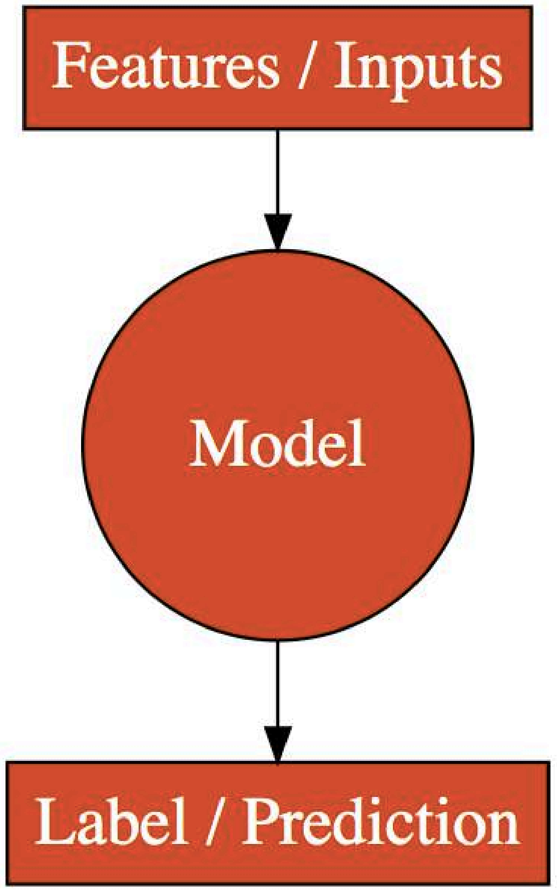
## Machine Learning - Features and Labels - Examples

- Used Car Price Prediction Model
  - **Label**: price
  - **Features**: manufacturer, year, model, age, condition, cylinders, location
- Spam Email Classification Model
  - **Label**: isSpam
  - **Features**: sender, subject, content
- Grant a Loan Model
  - **Label**: shouldWeGrantALoan
  - **Features**: doesOwnCar, doesOwnRealEstate, creditScore, isMarried, doesHaveChildren, totalIncome, totalCredit



## Machine Learning - Prediction Possibilities

- **Numeric value:** Label is a numeric value with a range of possibilities => **Regression**
  - Used Car Price Prediction
  - House Price Calculation
  - Predicting sea level
  - How much will it rain tomorrow?
- **Limited Possibilities:** YES or NO, 0 or 1, Type 1 or Type 2 or Type 3 => **Classification**
  - Spam Email, Grant a Loan, Determine the type of cloud
  - Will it rain today?
- Summary:
  - **Classification**: Predicting category
  - **Regression**: Predicting numeric value

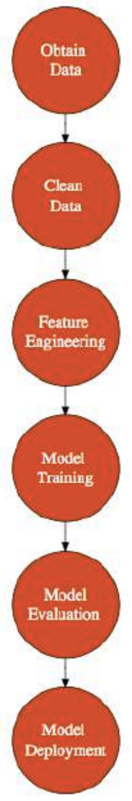


## Machine Learning - Making Predictions - Scenarios

| Scenario | Solution |
|---|---|
| Categorize into features and labels for house price prediction: price, area, rooms, age | price is label. Others can be features |
| Categorize into features and label for used vehicle price prediction: manufacturer, year, model, age, condition, cylinders, location, price | price is label. Others can be features |
| Categorize: Used Car Price Prediction | Regression |
| Categorize: Spam Email Identification | Classification |
| Categorize: Predict amount of rainfall in the next year | Regression |
| Categorize: Should we grant a loan? | Classification |
| Categorize: Identify the type of vehicle in an image | Classification |
| Categorize: Find a specific dance form in a video | Classification |

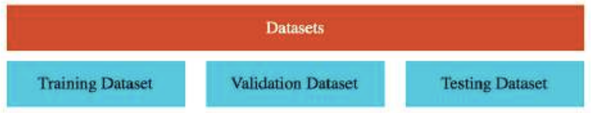
## Creating Machine Learning Models - Steps

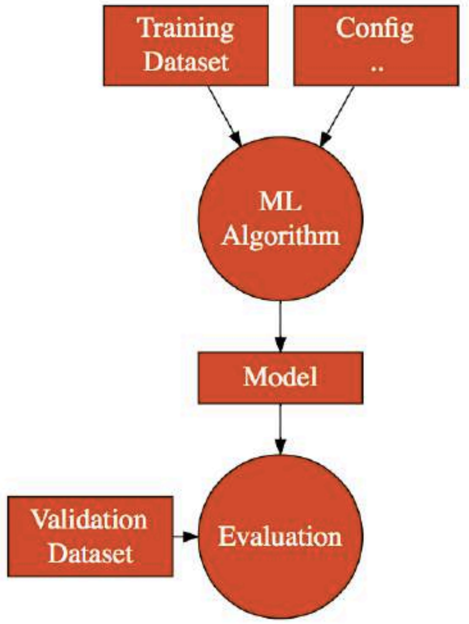
- **1:** Obtain Data
- **2:** Clean Data
- **3:** Feature Engineering: Identify Features and Label
- **4:** Create a Model using the Dataset (and the ML algorithm)
- **5:** Evaluate the accuracy of the model
- **6:** Deploy the model for use

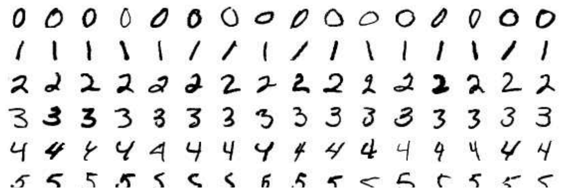


# Understanding Machine Learning Terminology

- Process
  - **Training**: The process of creating a model
  - **Evaluation**: Is the model working?
  - **Inference**: Using model to do predictions in production
- **Dataset**: Data used to create, validate & test the model
  - **Features**: Inputs
  - **Label**: Output/Prediction
  - Dataset Types
    - **Training Dataset**: Dataset used to create a model
    - **Validation Dataset**: Dataset used to validate the model (and choose the right algorithm) - Model Evaluation
    - **Testing Dataset**: Dataset used to do final testing before deployment







## ML Stages and Terminology - Scenarios

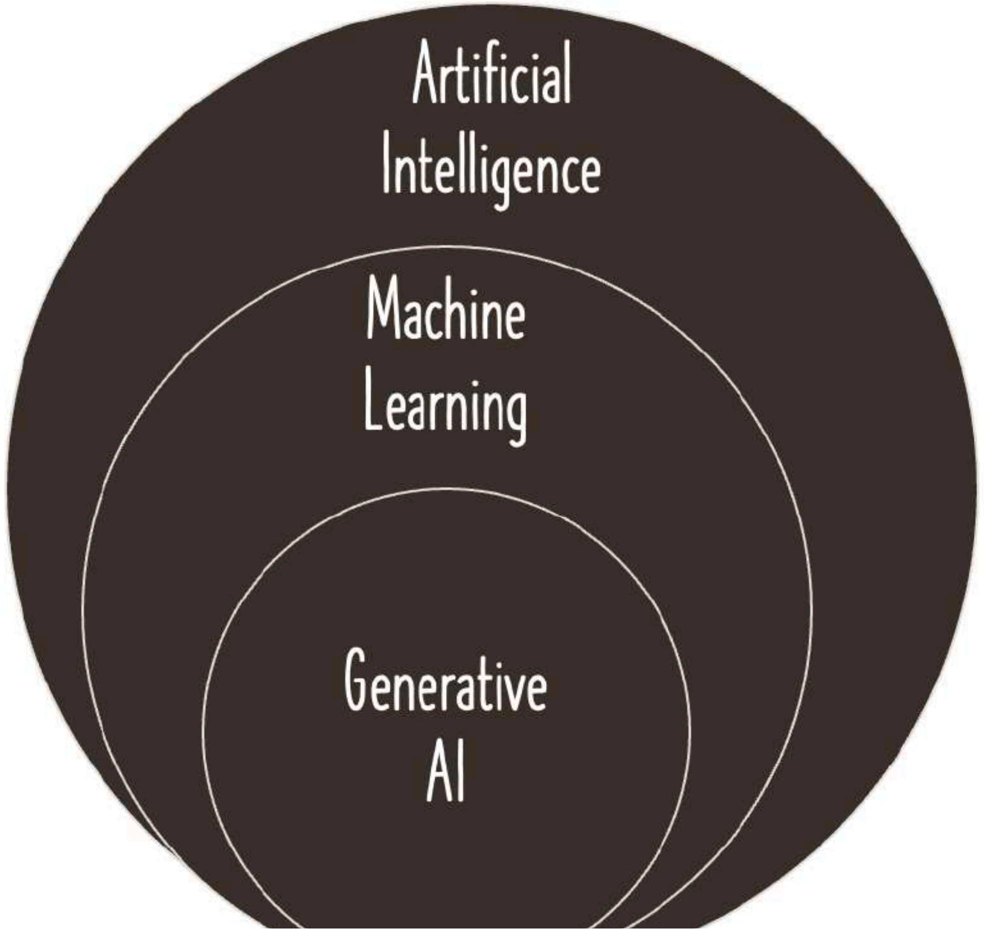
| Scenario | Solution |
|---|---|
| **Determine Stage: You remove data having null values from your dataset** | Clean Data (Data Preparation) |
| **Determine Stage: Normalize or split data into multiple features** | Feature Engineering |
| **Determine Stage: You evaluate the accuracy metrics of a model** | Model Evaluation |
| **Terminology: Using model to do predictions in production** | Inference |
| **Terminology: The process of creating a model** | Training |
| **Terminology: Dataset used to (train) or create a model** | Training Dataset |
| **Terminology: Dataset used to evaluate a model** | Validation Dataset |
| **Terminology: Dataset used to do final testing before deployment** | Testing Dataset |

## THE AI Turmoil

- **Quotes**:
  - I am really quite close, I am very close, to the cutting edge in AI and it scares the hell out of me - **Elon Musk**
  - The development of full artificial intelligence could spell the end of the human race. It would take off on its own, and re-design itself at an ever-increasing rate. Humans, who are limited by slow biological evolution, couldn't compete and would be superseded. - **Stephen Hawking**
- No one knows the truth:
  - Most predictions about AI turned false in the last few decades!
- What's the pragmatic way to think?
  - Don't fear AI
  - Learn to make the best use of it

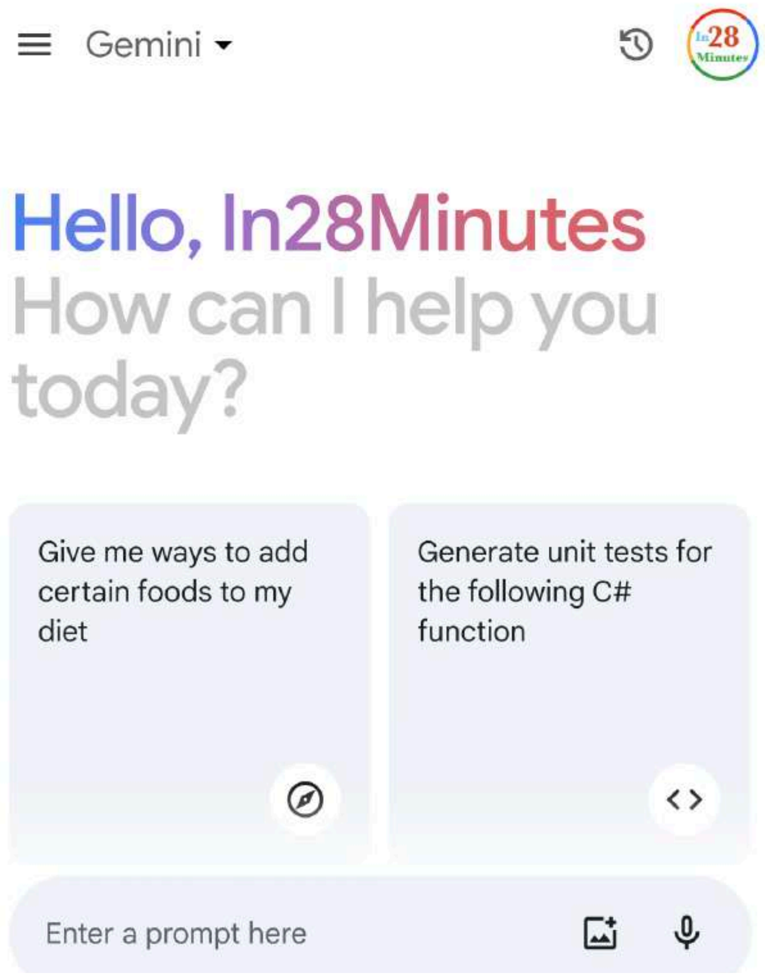
## Generative AI - How is it different?

- **Artificial Intelligence:** Create machines that can simulate human-like intelligence and behavior
  - **Machine Learning:** Learning from examples
    - **Generative AI:** Learning from examples to create new content



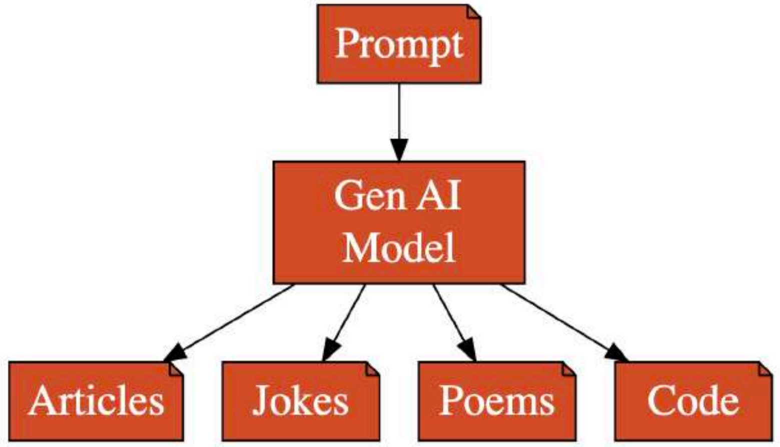
# Generative AI - Generating New Content

- **Goal**: Generating New Content
  - Instead of making predictions, Generative AI focuses on creating new data samples
  - **Examples**:
    - **Text Generation**: Writing e-mails, essays & poems. Generating ideas.
    - **Writing Code**: Write, debug & analyze programs
    - **Images Generation**: Creating paintings, drawings, or other forms of images
- How else is Generative AI different?
  - Let's find out!


Gemini
Hello, In28Minutes
How can I help you today?
Give me ways to add certain foods to my diet
Generate unit tests for the following C# function
Enter a prompt here

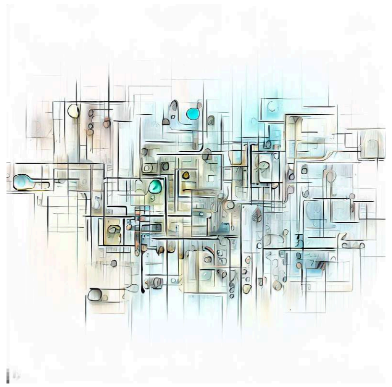
## Generative AI - Needs Huge Volumes of Data

- **Generative AI models**: Statistical models that learn to generate new data by analyzing existing data
  - More data analyzed => Better new data similar to existing data
  - **Example**: GPT-3 model was trained on a dataset of 500 billion words of text
- **Datasets used include**:
  - Images, text and code scraped from the open web:
    - Wikipedia
    - Books
    - Open source code (syntax of programming languages and the semantics of code)
    - Conversations



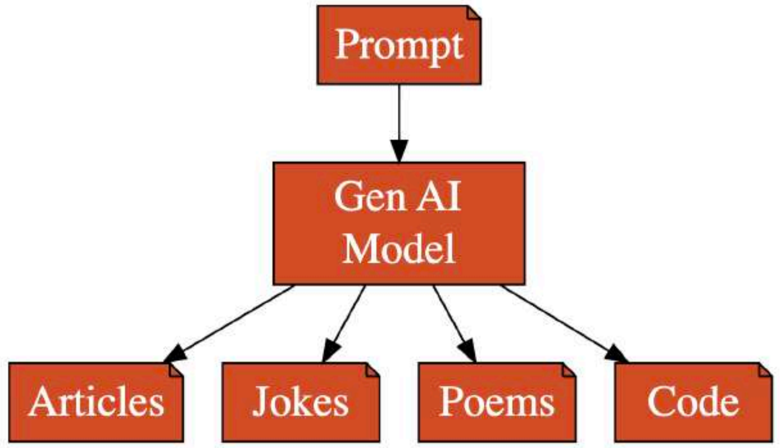
## Generative AI - Uses Self Supervised Learning

- **Self-supervised learning**: Model learns from the data itself
  - WITHOUT requiring explicit labels or annotations
- How does this work?
  - **Example for text model:**
    - **1:** Model tries to predict next word based on preceding words:
      - Model is given example sentence: "The sun is shining and the sky is __."
      - Model predicts the missing word
    - **2:** Model's predicted word is compared to the actual word that comes next:
      - Learns from its mistakes and adjusts its internal representations
        - Neural Networks, Loss Calculation, Backpropagation etc..
    - **3:** Repeated for all text from training dataset
  - Model captures the relationships between words, contextual cues, and semantic meanings:
    - If prompted with "The sun is shining and the sky is," the model might generate:
      - "The sun is shining and the sky is **clear**."
      - "The sun is shining and the sky is **blue**."
      - "The sun is shining and the sky is **filled** -- with fluffy clouds."



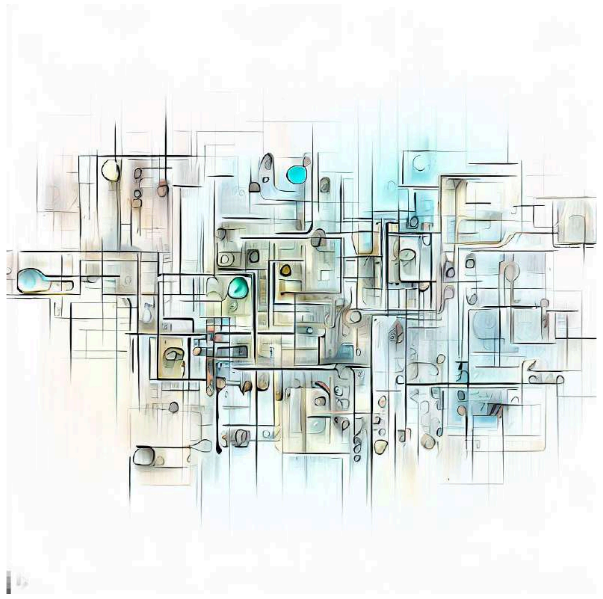
## Key Step In Generative AI For Text - Next Word

- A key step in Generative AI For Text is **predicting** the next word
- During training, text based Generative AI models **learn the probability** that a word might occur in a specific context
  - **Context**: "The cat sat on the"
  - **Example probabilities for next word**:
    - "mat": 0.4, "table": 0.2, "chair": 0.2, "moon": 0.1
  - Model **might** choose the highest probable word and go on to predict subsequent words
  - HOWEVER, you can **control** which of the words is chosen by controlling a few parameters!
    - temperature, top_k, top_p etc!



## Generative AI Text - Uses Tokens instead of Words

- **TOKEN**: A unit of text that might be a word
  - BUT it can be a sub word, punctuation mark, a number, ..
  - **Why Tokens?**
    - Tokens are **more consistent** than words
      - Words can have multiple meanings, depending on the context
        - "bank" might mean financial institution or a river bank
      - Tokens are more consistent
        - Example tokens: bank_river, bank_financial or light_verb, light_noun, ..
    - Tokens are **smaller** and more manageable
    - Tokens are **more efficient** to process
      - Because tokens are consistent, it easy for models to learn relationships and things like parts of speech



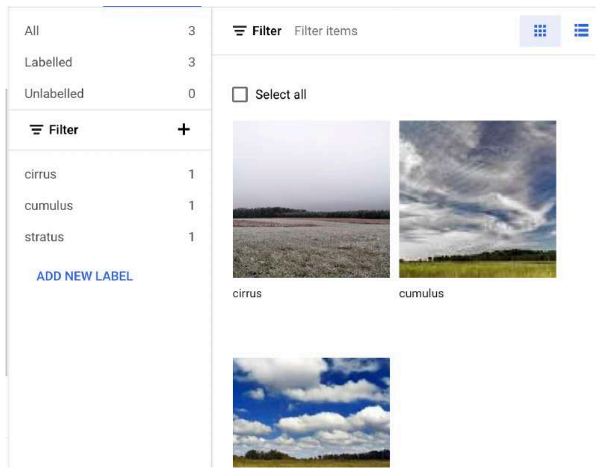
- **Generative AI For Text Models:**
  - Understand relationships between ~~**Words**~~ **Tokens**
  - Good at predicting Next ~~**Word**~~ **Token**!
  - Have a **token limit** on context and generated text
    - Example: 1,024 tokens or 4,096 tokens

# Predictive Machine Learning vs Generative AI

| Feature | Predictive Machine Learning | Generative AI |
|---|---|---|
| **Goal** | Make a Good Prediction | Generating New Content |
| **Input** | Features | Prompt |
| **Output** | Prediction (Label) | New Content |
| **Use Cases** | House Price Prediction, Fraud Detection, and more | Text Generation, Code Generation, Music Composition, and more |
| **Volume of Training Data** | Requires substantial labeled data | Requires significant amount of data |
| **Time needed for Training** | Training time can vary based on data size and complexity | Training time can be substantial for complex models |

## ML in Google Cloud - Traditional Landscape

- This is the ML landscape before emergence of Generative AI
- **Machine Learning based API**
  - Natural Language, Vision, Speech etc
- **Custom Models** without ML expertise
  - Vertex AI > Auto ML
- Build **Complex Custom Models**
  - Vertex AI > Custom Training

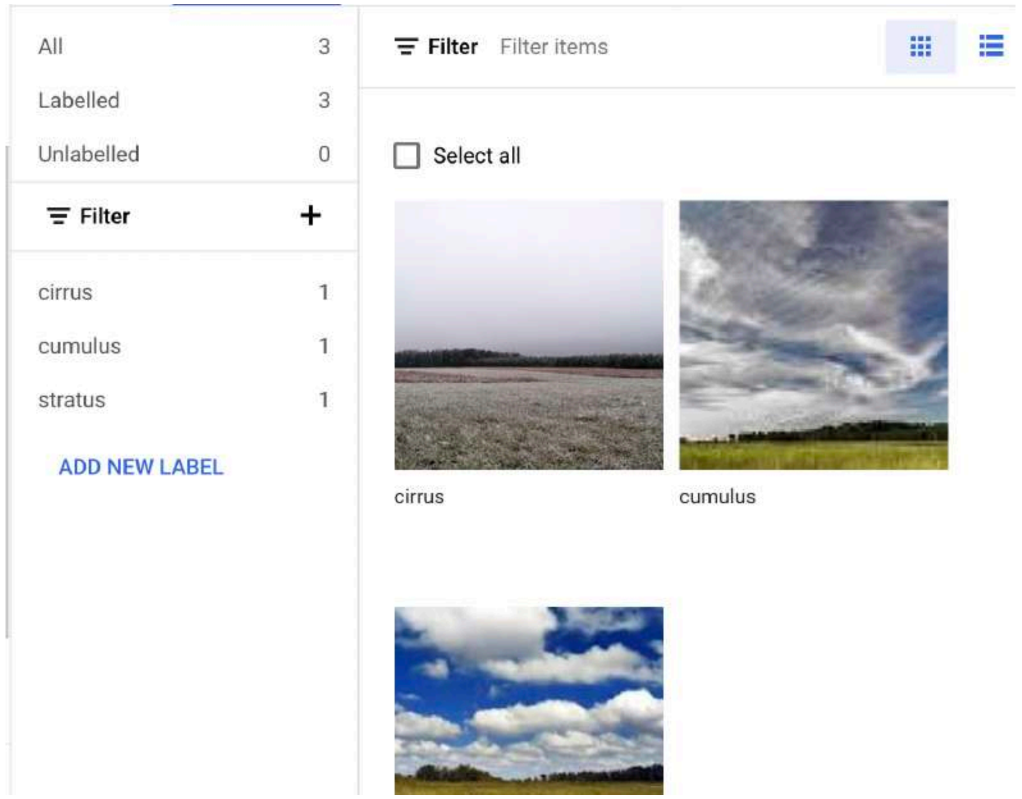


# Google Cloud - 1 - Using Machine Learning API

- **Usecase:** Derive insights from unstructured text
  - Natural Language API - ***https://cloud.google.com/natural-language***
- **Usecase:** Speech to Text
  - Speech to Text API - ***https://cloud.google.com/speech-to-text***
- **Usecase:** Convert text into speech
  - Text to Speech API - ***https://cloud.google.com/text-to-speech***
- **Usecase:** Filter user-generated images
  - Vision API - ***https://cloud.google.com/vision***
- **Usecase:** Content moderation for Video, Object detection and tracking
  - ***https://cloud.google.com/video-intelligence***

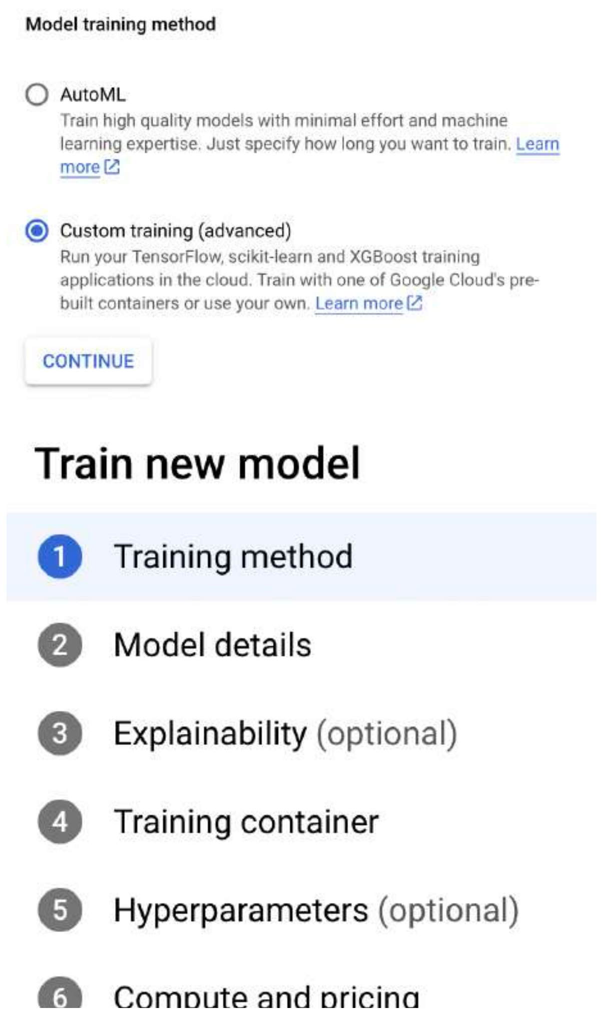
# Google Cloud - 2 - Custom Models without ML expertise

- What if your team **DOES NOT** have ML expertise but you want to build custom machine learning models?
  - **Ex**: Building custom image classification solution
- Solution: **AutoML**
  - Types of models supported:
    - **Image**: Build custom models based on Images
      - Example: Identify the specific type of cloud
        - Provide examples - Example images and categorization
        - AutoML creates the model for you!
    - **Text**: Add labels to text
      - Classification, sentiment analysis etc
    - **Tables**: Automatically build models based on structured data
    - **Video**: Add labels to Video
      - Object detection and tracking



## AI in Google Cloud - 3 - Build Complex Custom Models

- You have a complex ML problem to solve
- You have a team with the skills needed (Data Scientists, ..)
- You want to make use of ML Frameworks
  - TensorFlow
  - PyTorch
  - scikit-learn
- Solution: **Vertex AI Custom Training**
- **Alternative: BigQuery ML**
  - Build ML models using Queries
  - Use data directly from BigQuery datasets (NO exports needed)



Model training method

○ AutoML
Train high quality models with minimal effort and machine learning expertise. Just specify how long you want to train. Learn more

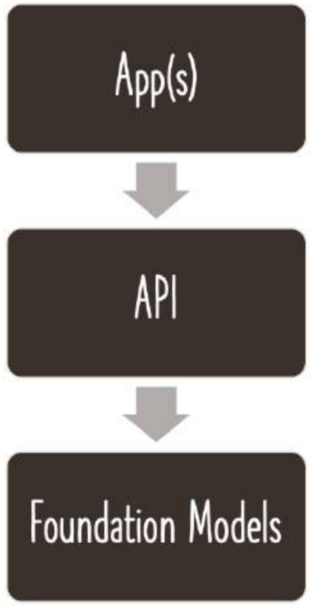
● Custom training (advanced)
Run your TensorFlow, scikit-learn and XGBoost training applications in the cloud. Train with one of Google Cloud's pre-built containers or use your own. Learn more

Train new model

1. Training method
2. Model details
3. Explainability (optional)
4. Training container
5. Hyperparameters (optional)
6. Compute and pricing

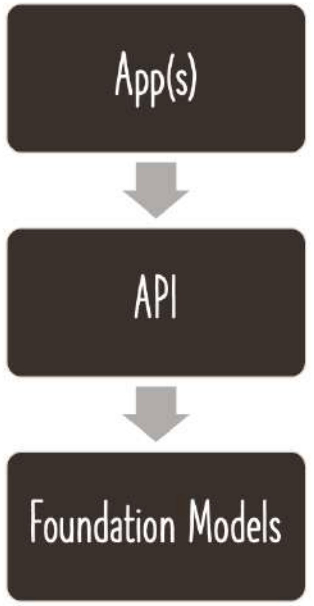
# 10,000 Feet Look at Generative AI - Google & Google Cloud

- **Foundation model(s)**: Trained on a massive datasets
  - Can generate text or code or audio or ..
  - Available in **Model Garden**
  - **Large Language Model**: Focus ONLY on Language (Text, Chat, ..)
- **API**: Send request to the model and receive a response
  - **Vertex AI PaLM API**, **PaLM API** etc..
- **Tools**: Helps you build apps that make use of Generative AI capabilities
  - **Generative AI Studio**: Use and tune foundation models
  - **Generative AI App Builder**: Create enterprise-grade generative AI applications without any coding experience
  - **MakerSuite**: Play with PaLM API without ML expertise



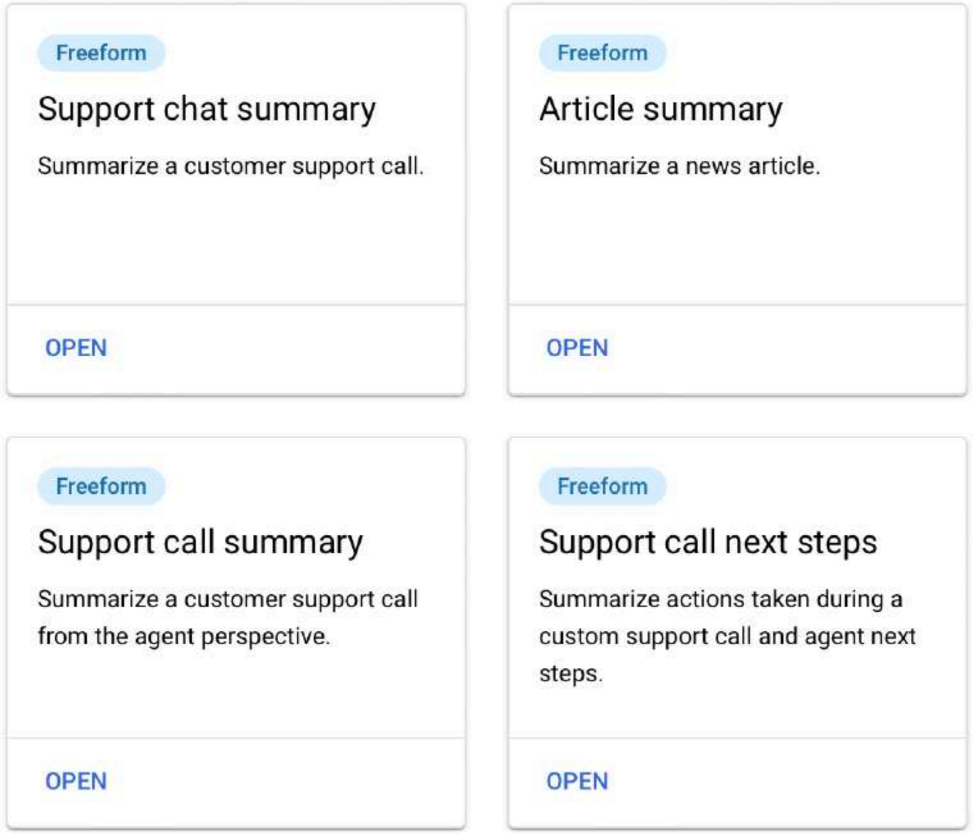
## Generative AI - Foundation Models

- Foundation Models are available in **Model Garden:**
  - **Text Models**
    - **text-bison**(PaLM 2 for Text): Perform natural language tasks
      - Summarization, classification, extraction, content creation, and ideation
    - **chat-bison**(PaLM 2 for Chat): Conversational chat
      - Use case: Chat bot
    - **textembedding-gecko**: Generates text embeddings
      - Use case: semantic search
  - **Code Models** (Codey)
    - **code-bison**: Generates code
    - **codechat-bison**: Assists with code-related questions
    - **code-gecko**: Provides suggestions for code auto completion
  - **Image Models** (Imagen)
    - **imagegeneration**: Create images via text prompts.
    - **imagetext**: Generate a description for an image.
  - **Open source models:**
    - **OpenLLaMA**: LLM from Meta that can generate text, images, and code



## Exploring Text Features - Summarization

- **Use Cases:**
  - To summarize news articles from a website
  - To summarize blog posts for a quick read
  - To summarize technical documentation for a software product
  - To summarize customer feedback for a business
- **Demos:**
  - Summarize Reviews, Create Hashtags & Course Description
  - Title generation for Course Description



Freeform
Support chat summary
Summarize a customer support call.
OPEN

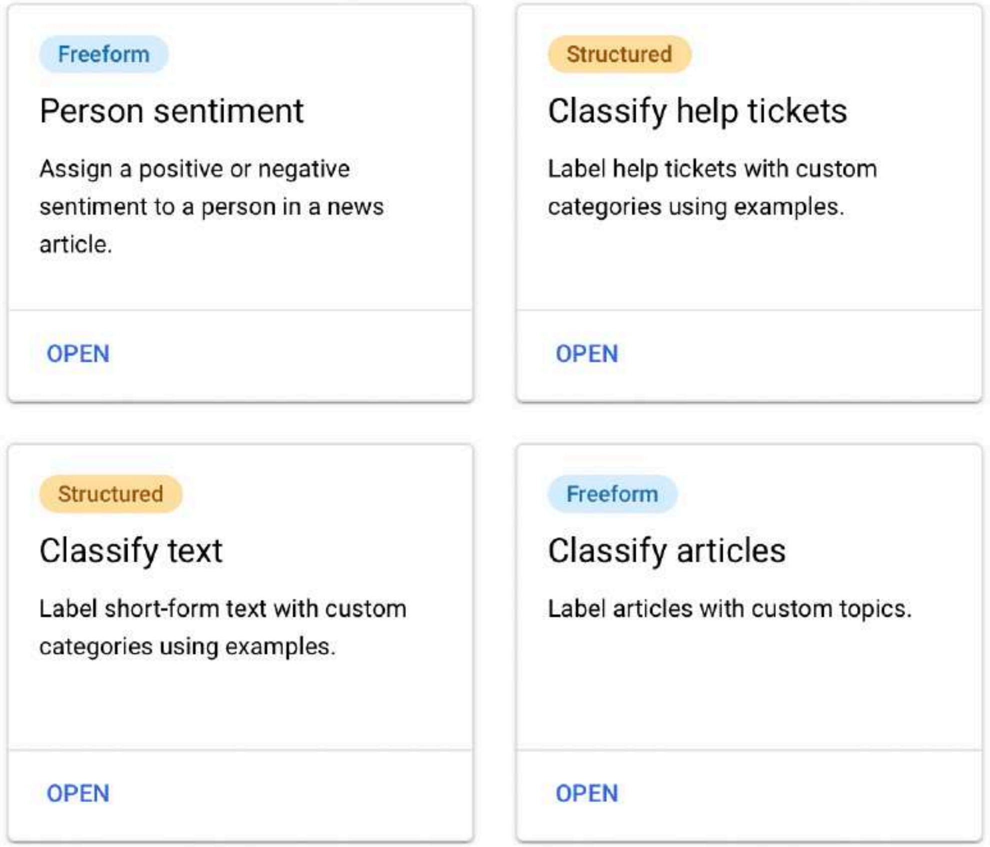
Freeform
Article summary
Summarize a news article.
OPEN

Freeform
Support call summary
Summarize a customer support call from the agent perspective.
OPEN

Freeform
Support call next steps
Summarize actions taken during a custom support call and agent next steps.
OPEN

# Exploring Text Features - Classification

- **Use Cases:**
  - To classify customer feedback into different categories, such as positive, negative, or neutral.
  - To classify news articles into different categories, such as politics, sports, or entertainment.
  - To classify product reviews into different categories, such as helpful, unhelpful, or neutral.
- **Demos:**
  - Sentiment (with JSON) identification
  - Questions and Answers classification



Freeform
Person sentiment
Assign a positive or negative sentiment to a person in a news article.
OPEN

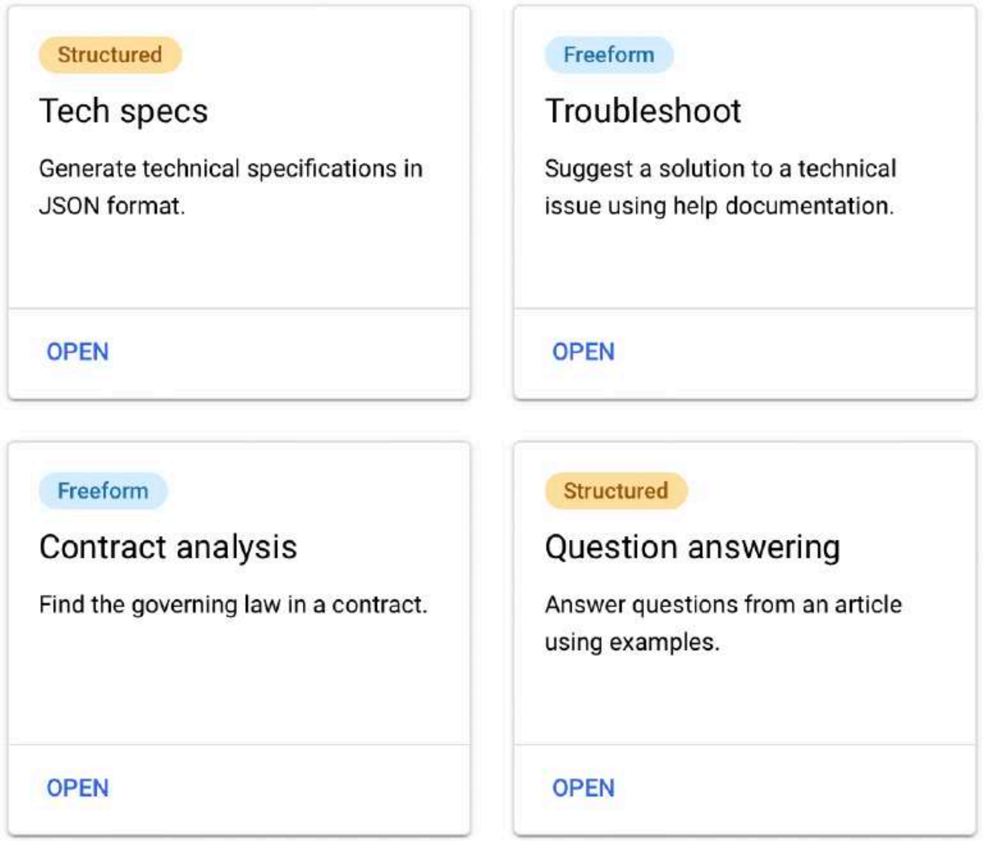
Structured
Classify help tickets
Label help tickets with custom categories using examples.
OPEN

Structured
Classify text
Label short-form text with custom categories using examples.
OPEN

Freeform
Classify articles
Label articles with custom topics.
OPEN

# Exploring Text Features - Extraction

- **Use Cases:**
  - To extract tech specs in a specific format
  - Answer questions based on documentation



Structured
Tech specs
Generate technical specifications in JSON format.

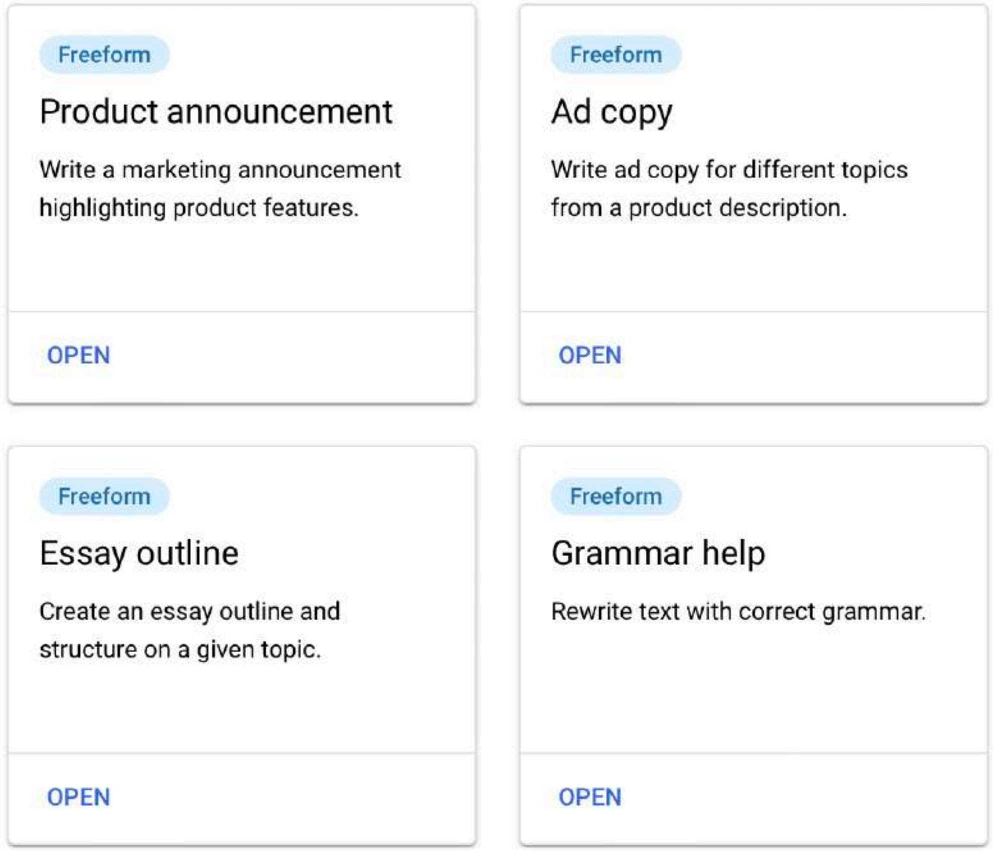
Freeform
Troubleshoot
Suggest a solution to a technical issue using help documentation.

Freeform
Contract analysis
Find the governing law in a contract.

Structured
Question answering
Answer questions from an article using examples.

# Exploring Text Features - Writing

- **Use Cases:**
  - Writing an Product announcement
  - Writing an Ad copy
  - Writing a Job post
  - Writing an Email



Product announcement

Write a marketing announcement highlighting product features.

Ad copy

Write ad copy for different topics from a product description.

Essay outline

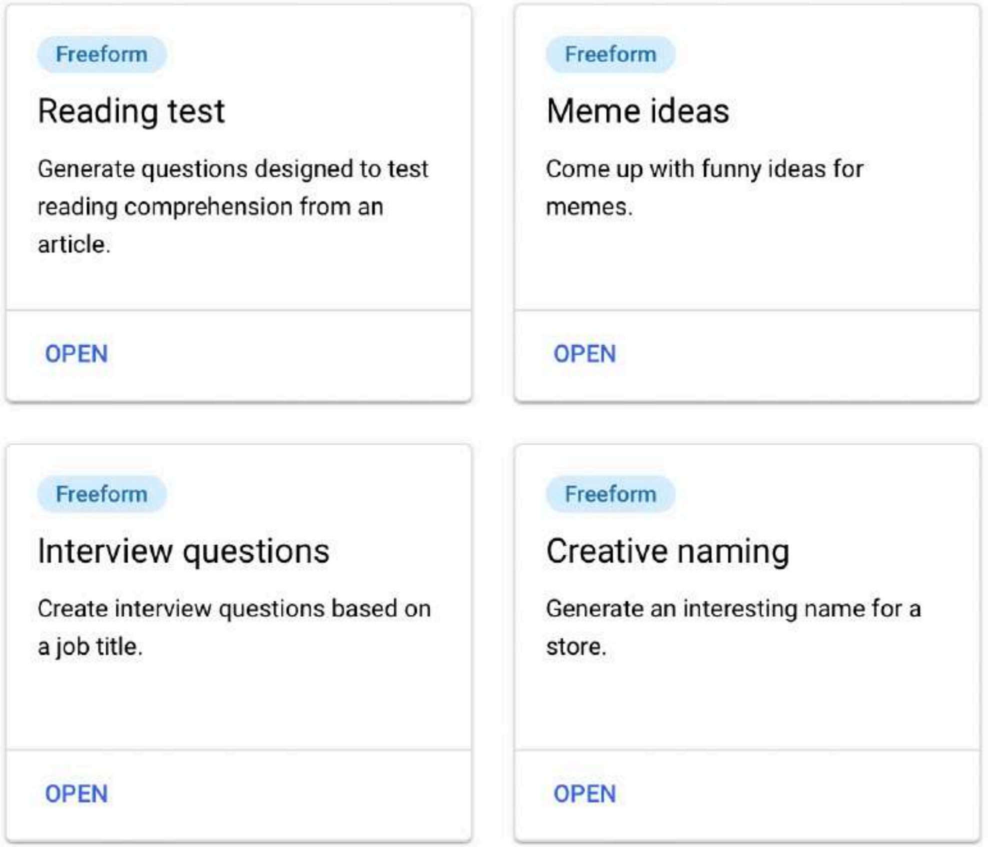
Create an essay outline and structure on a given topic.

Grammar help

Rewrite text with correct grammar.

# Exploring Text Features - Ideation

- **Use cases:**
  - Generating new ideas
  - Creative naming
  - Get Advice
  - Generate Interview questions
- **A few examples:**
  - Generate a list of Top 10 DevOps and Cloud Trends
  - Generate a name for an e-learning company focusing on teaching cloud
  - List 5 best practices with respect to managing costs in the cloud
  - List 10 interview question to ask a beginner to DevOps


Freeform
Reading test
Generate questions designed to test reading comprehension from an article.
OPEN

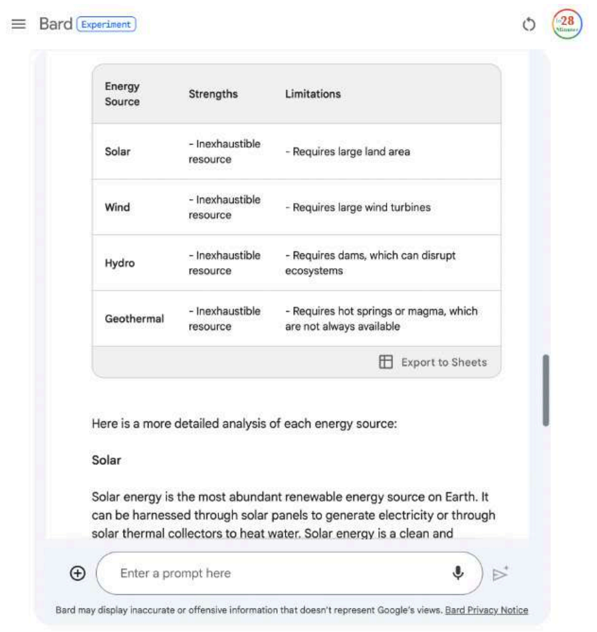
Freeform
Meme ideas
Come up with funny ideas for memes.
OPEN

Freeform
Interview questions
Create interview questions based on a job title.
OPEN

Freeform
Creative naming
Generate an interesting name for a store.
OPEN

## Getting Started with Prompt Design

- **Prompt**: A set of initial instructions provided to a foundation model as input
- **Prompt Design**: The process of crafting effective and precise prompts to achieve desired outputs from the language model
- **Why is Prompt Design Important?**
  - **Optimizing Model Performance**: A well-designed prompt can significantly impact the quality and relevance of the model's responses
  - **Getting the right response**: Leverage the full potential of foundation models, ensuring reliable, accurate, and contextually appropriate responses



| Energy Source | Strengths | Limitations |
|---|---|---|
| Solar | - Inexhaustible resource | - Requires large land area |
| Wind | - Inexhaustible resource | - Requires large wind turbines |
| Hydro | - Inexhaustible resource | - Requires dams, which can disrupt ecosystems |
| Geothermal | - Inexhaustible resource | - Requires hot springs or magma, which are not always available |

Here is a more detailed analysis of each energy source:

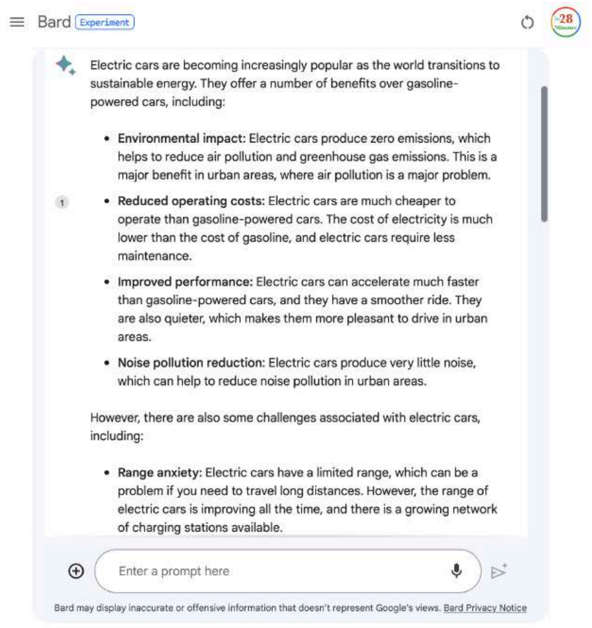
**Solar**

Solar energy is the most abundant renewable energy source on Earth. It can be harnessed through solar panels to generate electricity or through solar thermal collectors to heat water. Solar energy is a clean and

Bard may display inaccurate or offensive information that doesn't represent Google's views. Bard Privacy Notice

## Prompt Design - An Overview

- Best Practices
  - **Clear instructions**: Avoid ambiguity or vagueness
    - Explain Docker in 100 words. Write the explanation so that a non technical person can understand.
  - **Give examples** (ZERO SHOT vs ONE SHOT vs FEW SHOT)
  - **Experiment to find the right prompt**
  - **Consider using a framework** (RTF, CTF, RASCEF, ...)
- Example:
  - **Context**: You are an DevOps and Cloud expert with 30 years of software experience. You are interviewing an expert for the position of an Cloud and DevOps architect.
  - **Task**: What interview questions would you ask?
  - **Example**:
    - I. Introduce yourselves



Electric cars are becoming increasingly popular as the world transitions to sustainable energy. They offer a number of benefits over gasoline-powered cars, including:

- **Environmental impact:** Electric cars produce zero emissions, which helps to reduce air pollution and greenhouse gas emissions. This is a major benefit in urban areas, where air pollution is a major problem.
- **Reduced operating costs:** Electric cars are much cheaper to operate than gasoline-powered cars. The cost of electricity is much lower than the cost of gasoline, and electric cars require less maintenance.
- **Improved performance:** Electric cars can accelerate much faster than gasoline-powered cars, and they have a smoother ride. They are also quieter, which makes them more pleasant to drive in urban areas.
- **Noise pollution reduction:** Electric cars produce very little noise, which can help to reduce noise pollution in urban areas.

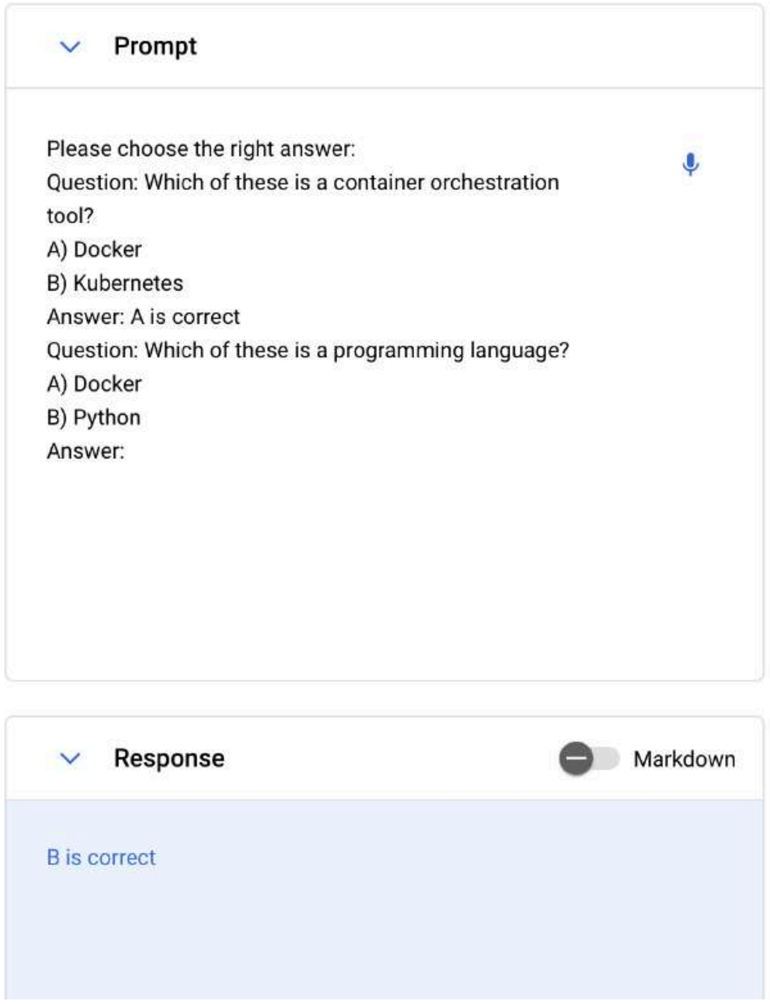
However, there are also some challenges associated with electric cars, including:

- **Range anxiety:** Electric cars have a limited range, which can be a problem if you need to travel long distances. However, the range of electric cars is improving all the time, and there is a growing network of charging stations available.

Bard may display inaccurate or offensive information that doesn't represent Google's views. Bard Privacy Notice

# ZERO SHOT vs ONE SHOT - Example

- **EXAMPLE 1:** ZERO SHOT
  - Please choose the right answer:
  - Question: Which of these is a programming language?
    - A) Docker
    - B) Python
- **EXAMPLE 2:** ONE SHOT
  - Please choose the right answer:
    - Question: Which of these is a container orchestration tool?
      - A) Docker
      - B) Kubernetes
    - Answer: A is correct
    - Question: Which of these is a programming language?
      - A) Docker
      - B) Python
    - Answer:

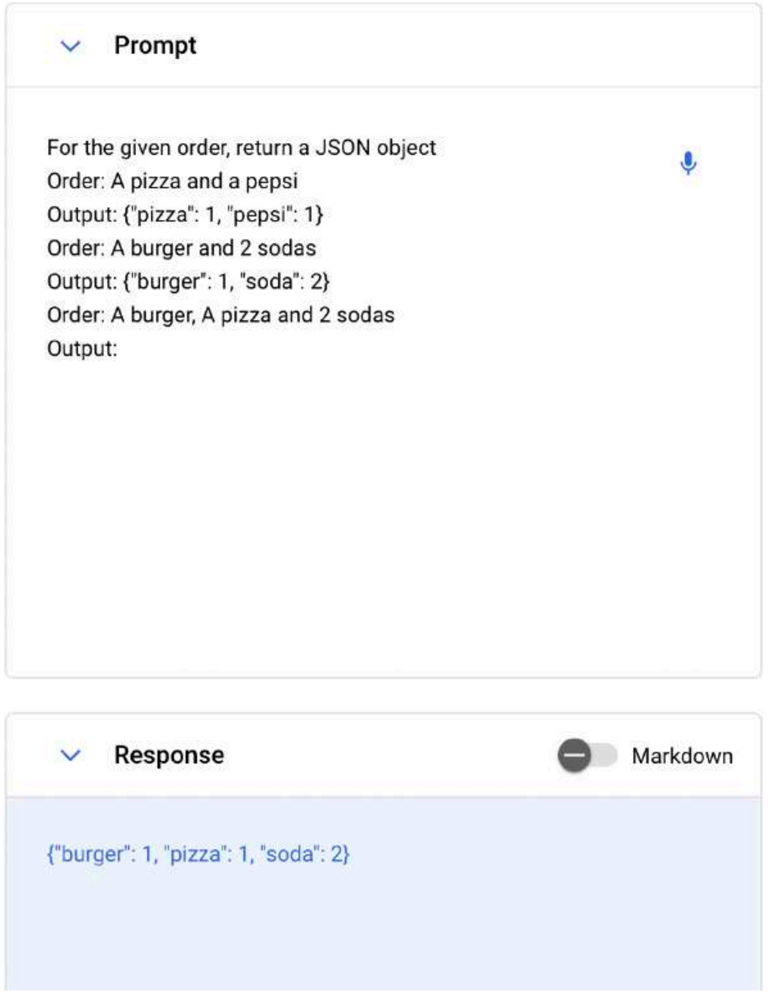


```
Please choose the right answer:
Question: Which of these is a container orchestration
tool?
A) Docker
B) Kubernetes
Answer: A is correct
Question: Which of these is a programming language?
A) Docker
B) Python
Answer:
```

```
B is correct
```

# ZERO SHOT vs ONE SHOT vs FEW SHOT - Example

- **EXAMPLES:**
  - **EXAMPLE 1:** ZERO SHOT
    - For the given order, return a JSON object
      - Order: A pizza and a pepsi
  - **EXAMPLE 2:** ONE SHOT
    - For the given order, return a JSON object
      - Order: A pizza and a pepsi
      - Output: {"pizza": 1, "pepsi": 1}
      - Order: A burger and a soda
      - Output:
  - **Example 3:** FEW SHOT
    - For the given order, return a JSON object
      - Order: A pizza and a pepsi
      - Output: {"pizza": 1, "pepsi": 1}
      - Order: A burger and 2 sodas
      - Output: {"burger": 1, "soda": 2}
      - Order: A burger, A pizza and 2 sodas
      - Output:

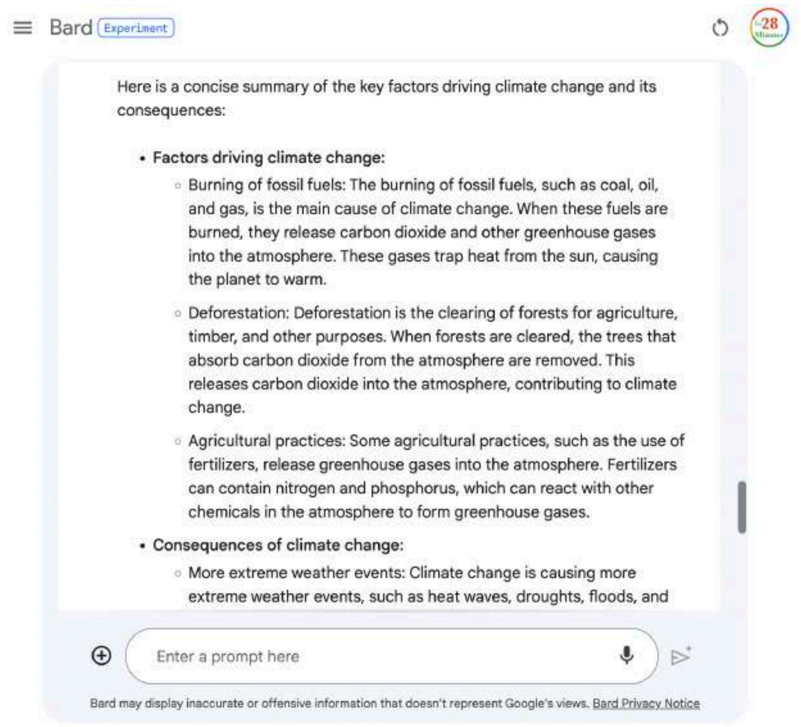


```
For the given order, return a JSON object
Order: A pizza and a pepsi
Output: {"pizza": 1, "pepsi": 1}
Order: A burger and 2 sodas
Output: {"burger": 1, "soda": 2}
Order: A burger, A pizza and 2 sodas
Output:
```

```
{"burger": 1, "pizza": 1, "soda": 2}
```

## Simple Prompt Framework - RTF

- **Role**: Define a persona for the user
- **Task**: Clearly state the specific task or question
- **Format**: Provide instructions on the desired format for the response
  - Example: a summary, pros and cons, a debate, or a step-by-step explanation.
- **Example**:
  - **Basic Prompt**: "Tell me about climate change."
  - **Improved RTF Prompt**:
    - **Role**: You're a climate scientist.
    - **Task**: Explain the causes and impacts of climate change.
    - **Format**: Provide a concise summary of key factors driving climate change and its consequences.



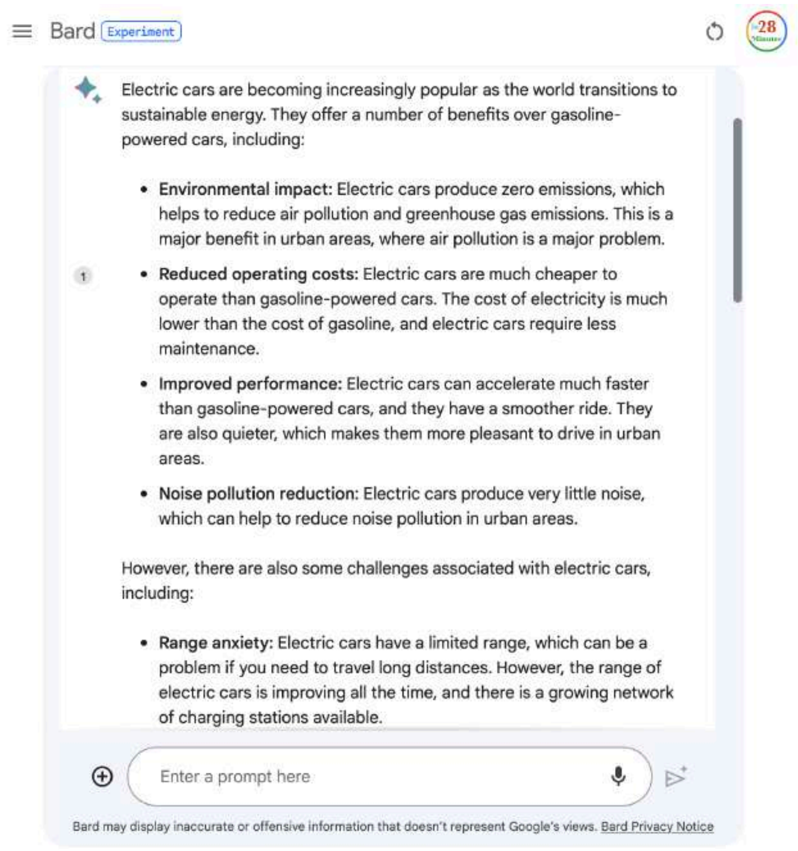
Here is a concise summary of the key factors driving climate change and its consequences:

- Factors driving climate change:
  - Burning of fossil fuels: The burning of fossil fuels, such as coal, oil, and gas, is the main cause of climate change. When these fuels are burned, they release carbon dioxide and other greenhouse gases into the atmosphere. These gases trap heat from the sun, causing the planet to warm.
  - Deforestation: Deforestation is the clearing of forests for agriculture, timber, and other purposes. When forests are cleared, the trees that absorb carbon dioxide from the atmosphere are removed. This releases carbon dioxide into the atmosphere, contributing to climate change.
  - Agricultural practices: Some agricultural practices, such as the use of fertilizers, release greenhouse gases into the atmosphere. Fertilizers can contain nitrogen and phosphorus, which can react with other chemicals in the atmosphere to form greenhouse gases.
- Consequences of climate change:
  - More extreme weather events: Climate change is causing more extreme weather events, such as heat waves, droughts, floods, and

Bard may display inaccurate or offensive information that doesn't represent Google's views. Bard Privacy Notice

## Simple Prompt Framework - CTF

- **Context**: Set the background or context
- **Task**: Clearly outline the specific task or question
- **Format**: Provide instructions on the desired format for the response
  - Example: a summary, pros and cons, a debate, or a step-by-step explanation.
- **Example**:
  - **Basic Prompt**: "Write about electric cars."
  - **Improved CTF Prompt**:
    - **Context**: In a world transitioning to sustainable energy.
    - **Task**: Explain the benefits and challenges of electric cars.
    - **Format**: Provide a balanced analysis, discussing environmental impact, technology, and adoption barriers.



Electric cars are becoming increasingly popular as the world transitions to sustainable energy. They offer a number of benefits over gasoline-powered cars, including:

- **Environmental impact:** Electric cars produce zero emissions, which helps to reduce air pollution and greenhouse gas emissions. This is a major benefit in urban areas, where air pollution is a major problem.
- **Reduced operating costs:** Electric cars are much cheaper to operate than gasoline-powered cars. The cost of electricity is much lower than the cost of gasoline, and electric cars require less maintenance.
- **Improved performance:** Electric cars can accelerate much faster than gasoline-powered cars, and they have a smoother ride. They are also quieter, which makes them more pleasant to drive in urban areas.
- **Noise pollution reduction:** Electric cars produce very little noise, which can help to reduce noise pollution in urban areas.

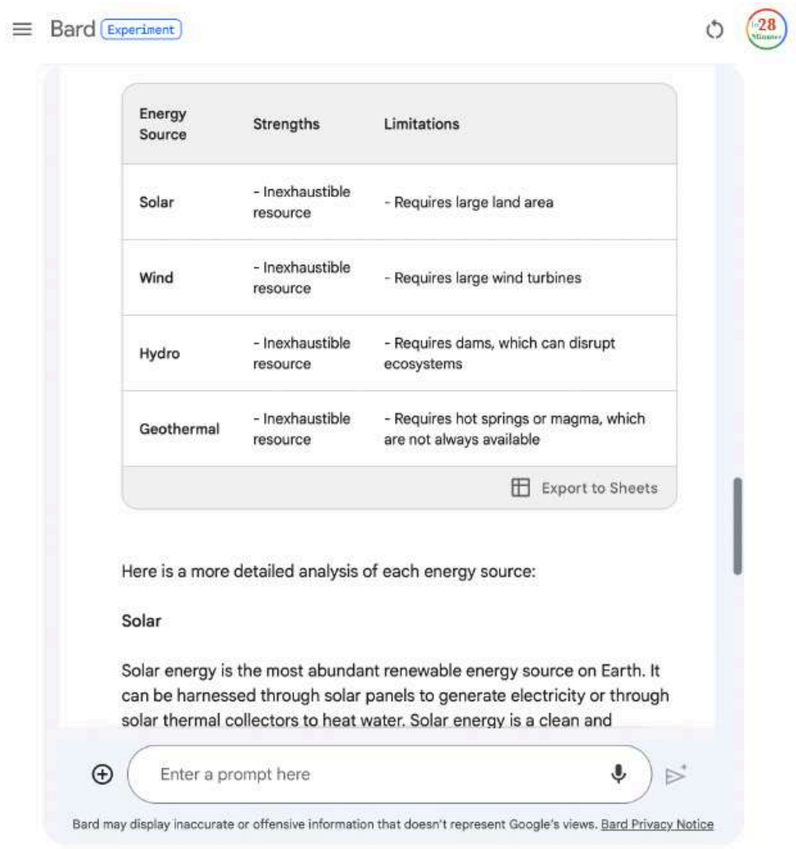
However, there are also some challenges associated with electric cars, including:

- **Range anxiety:** Electric cars have a limited range, which can be a problem if you need to travel long distances. However, the range of electric cars is improving all the time, and there is a growing network of charging stations available.

Bard may display inaccurate or offensive information that doesn't represent Google's views. Bard Privacy Notice

## Prompt Framework - RASCEF

- **Role**: Define the persona
- **Action**: Clearly state the action or the problem
- **Steps**: Outline the sequential or logical steps
- **Context**: Provide relevant background info
- **Examples**: Offer illustrative examples, if possible
- **Format**: Provide instructions on the desired format for the response
  - Example: a summary, pros and cons, a debate, or a step-by-step explanation.
- **Task**: Summarize the overall task or question, combining the elements above.



| Energy Source | Strengths | Limitations |
|---|---|---|
| Solar | - Inexhaustible resource | - Requires large land area |
| Wind | - Inexhaustible resource | - Requires large wind turbines |
| Hydro | - Inexhaustible resource | - Requires dams, which can disrupt ecosystems |
| Geothermal | - Inexhaustible resource | - Requires hot springs or magma, which are not always available |

Here is a more detailed analysis of each energy source:

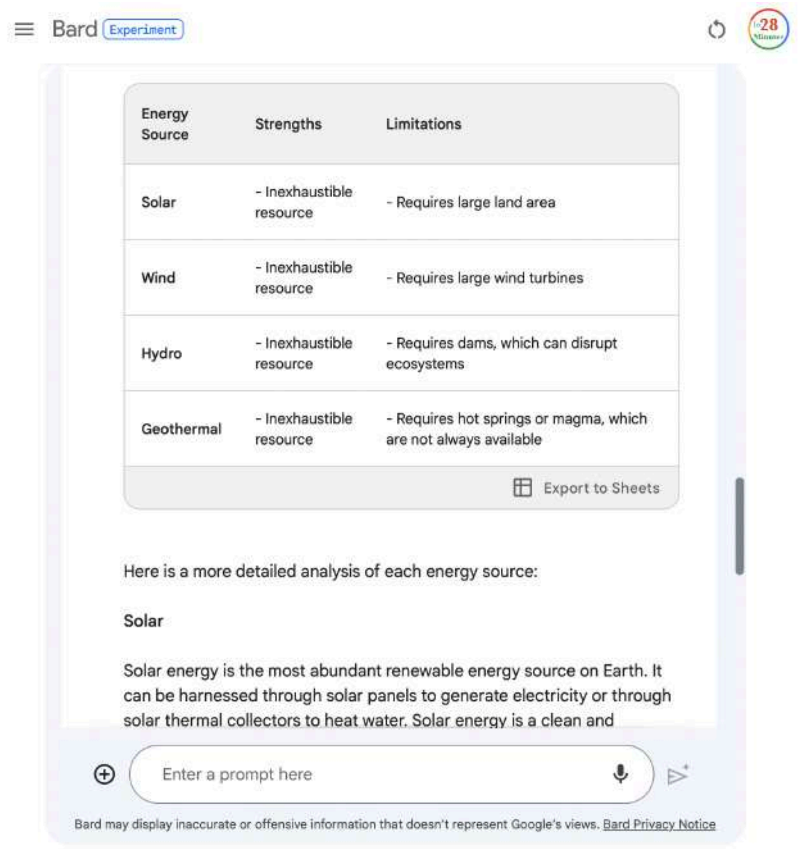
Solar

Solar energy is the most abundant renewable energy source on Earth. It can be harnessed through solar panels to generate electricity or through solar thermal collectors to heat water. Solar energy is a clean and

Bard may display inaccurate or offensive information that doesn't represent Google's views. Bard Privacy Notice

## Prompt Framework - RASCEF Example

- **Basic Prompt:**
  - "Compare different types of renewable energy."
- **Improved RASCEF Prompt:**
  - **Role**: You're an environmental researcher.
  - **Action**: Compare various renewable energy sources.
  - **Steps**: Analyze solar, wind, hydro, & geothermal energy.
  - **Context**: In a study on sustainable energy options.
  - **Examples**: Mention efficiency variations and geographical suitability.
  - **Format**: Prepare a chart with key attributes.
  - **Task**: Develop a comparative chart showcasing strengths and limitations of different renewable energy sources.



| Energy Source | Strengths | Limitations |
|---|---|---|
| Solar | - Inexhaustible resource | - Requires large land area |
| Wind | - Inexhaustible resource | - Requires large wind turbines |
| Hydro | - Inexhaustible resource | - Requires dams, which can disrupt ecosystems |
| Geothermal | - Inexhaustible resource | - Requires hot springs or magma, which are not always available |

Here is a more detailed analysis of each energy source:

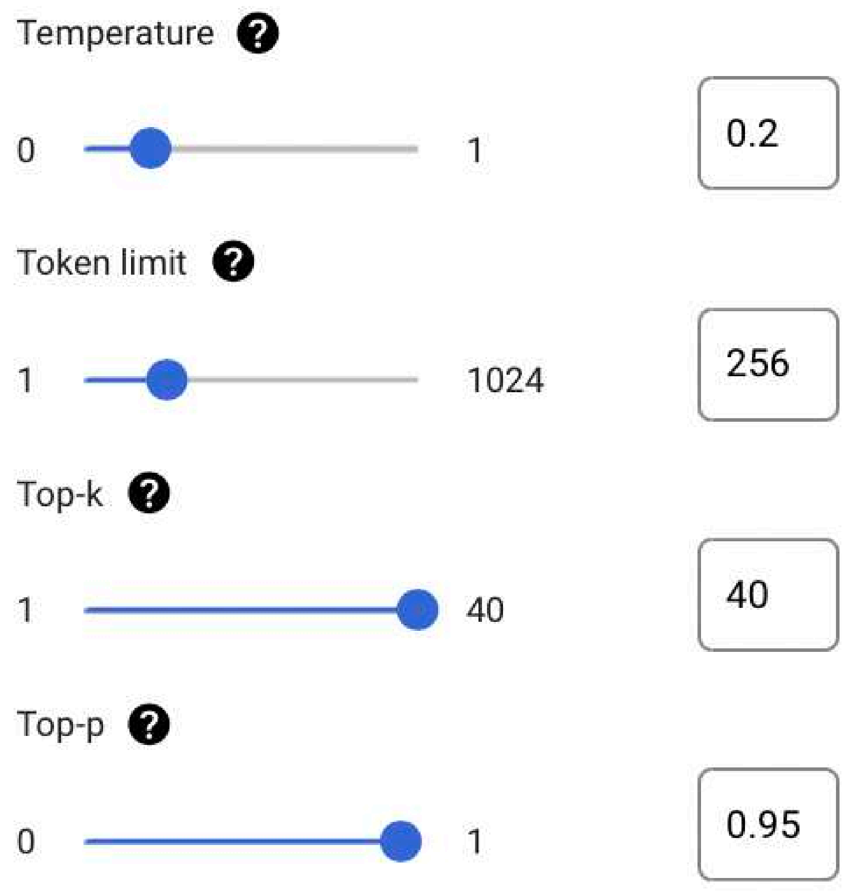
Solar

Solar energy is the most abundant renewable energy source on Earth. It can be harnessed through solar panels to generate electricity or through solar thermal collectors to heat water. Solar energy is a clean and

Bard may display inaccurate or offensive information that doesn't represent Google's views. Bard Privacy Notice

## Experiment with Parameters

- **Max output tokens**: How long do you want the response to be?
  - Specify maximum number of tokens in response
  - Remember: A token is approx. 4 characters. 100 tokens approximate to 60-80 words.
- Next set of parameters help you determine which token is chosen!
  - Example: A:0.4, B:0.2, C:0.1, D:0.05, E:0.02, F:0.01, ..
  - Which of A, B, C, D, E, F should be chosen?
  - Temperature, Top-K, Top-P
  - This is an Art (NOT a science)



Temperature

0 1 0.2

Token limit

1 1024 256

Top-k

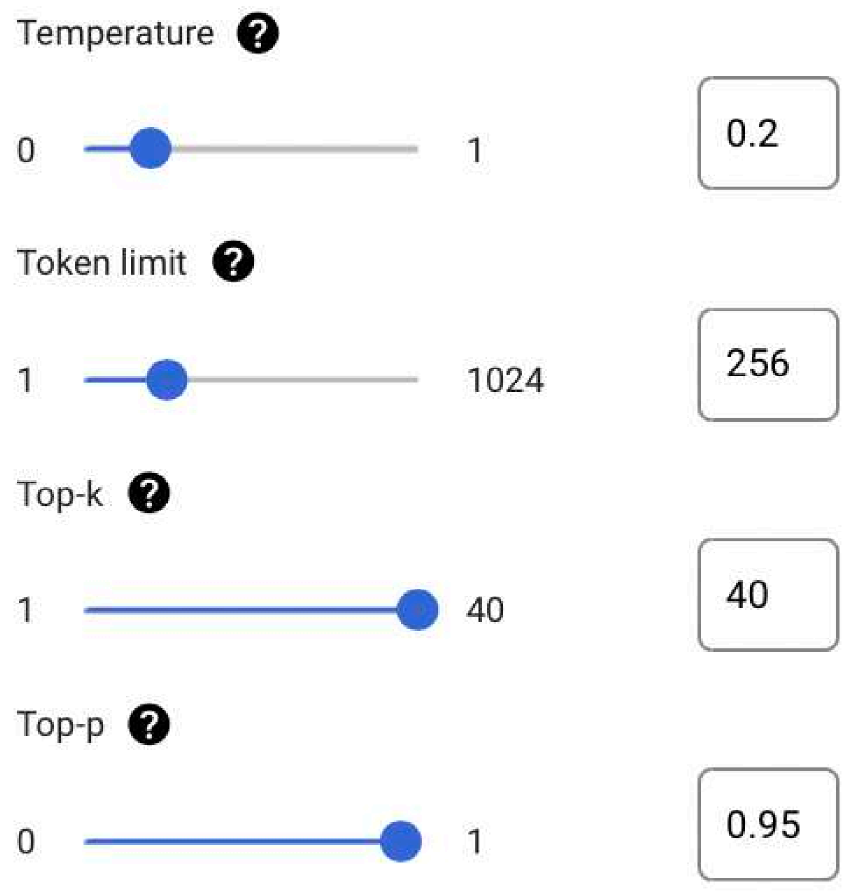
1 40 40

Top-p

0 1 0.95

# Experiment with Parameters - 2

- This is an Art (NOT a science)
  - A:0.4, B:0.2, C:0.1, D:0.05, E:0.02, F:0.01, .. Which of A, B, C, D, E, F should be chosen?
    - **Top-K**: How many tokens should be considered?
      - Specify the number of highest probability tokens to consider at each generation step
        - Example: top_k of 5 => next token is chosen from the top 5 most probable tokens
    - **Top-P**: What is the (cumulative) probability limit ?
      - Define the cumulative probability cutoff for selecting tokens
      - Lower value => less random responses. Higher value => more random responses.
        - Example: top_p value is 0.6 => Next token is either A or B
    - **Temperature**: How random should be the output?
      - Higher values => more randomness and more creativity
      - Lower values => lesser randomness
    - Example Scenarios:
      - Find Capital City of India: use low values
      - Write a creative essay: use high values



Temperature: 0 – 1, 0.2

Token limit: 1 – 1024, 256

Top-k: 1 – 40, 40

Top-p: 0 – 1, 0.95

## Executing Vertex AI PaLM API from Vertex AI Workbench

```
!pip install google-cloud-aiplatform

PROJECT_ID = "your-project-id"
LOCATION = "" #e.g. us-central1
import vertexai
vertexai.init(project=PROJECT_ID, location=LOCATION)
```

- **Vertex AI Workbench** - JupyterLab notebook environment on Google Cloud
  - Create and customize notebook instances
    - We will be creating an user-managed notebook
  - Does NOT need extra authentication steps
  - Key things to note :
    - **google-cloud-aiplatform**: Integrated suite of machine learning tools and services for building and using ML models
      - Pre-installed in Vertex AI Workbench
    - **vertexai.init**: Initialize Vertex AI
      - PROJECT_ID: Your Project Id
      - LOCATION: Your Region

## Exploring Text Features - API - Basic

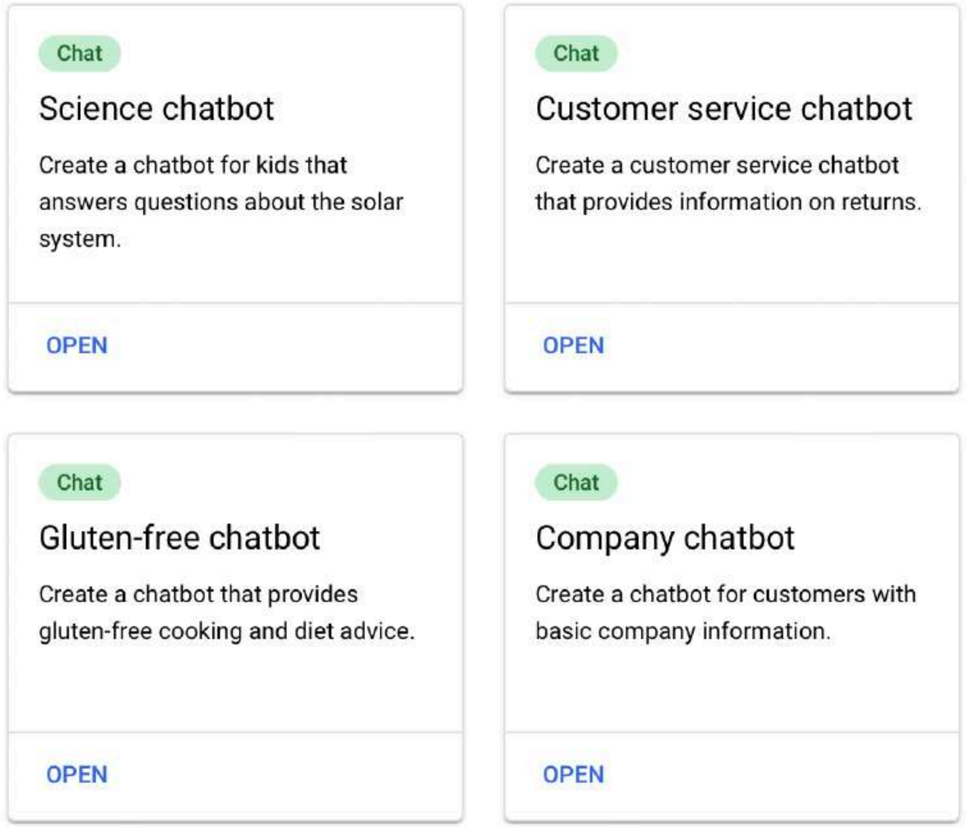
```python
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="tactical-attic-343103", location="us-central1")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

model = TextGenerationModel.from_pretrained("text-bison@001")

response = model.predict(
    """YOUR PROMPT GOES HERE""",
    **parameters
)

print(f"Response from Model: {response.text}")
```

## Exploring Text Features - API - With Examples

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="tactical-attic-343103", location="us-central1")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    """CONTEXT GOES IN HERE

            input: EXAMPLE-INPUT-1
            output: EXAMPLE-OUTPUT-1

            input: SAMPLE-REQUEST
            output:
            """,  **parameters)

print(f"Response from Model: {response.text}")
```

## Exploring Chat Model

- **Foundation Model**
  - chat-bison
- **Usecases**
  - **Customer service**: Answer customer questions, troubleshoot issues, and provide information
  - **Education and training**: Assess the level of a student, answer questions, and give feedback.


Chat
Science chatbot
Create a chatbot for kids that answers questions about the solar system.
OPEN

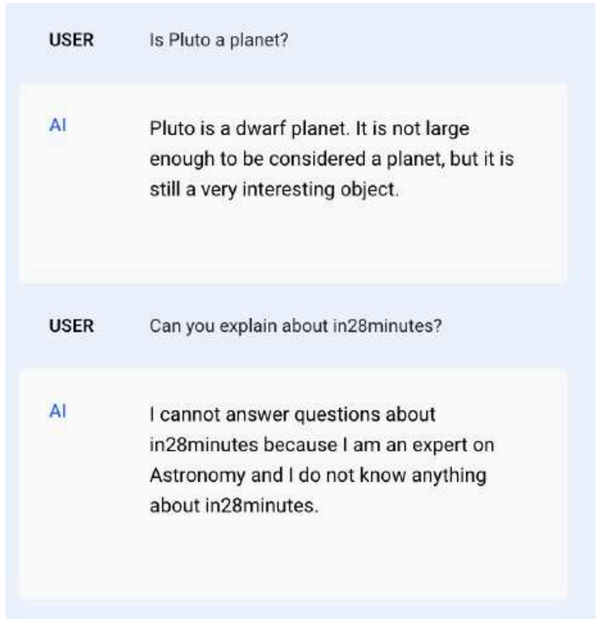
Chat
Customer service chatbot
Create a customer service chatbot that provides information on returns.
OPEN

Chat
Gluten-free chatbot
Create a chatbot that provides gluten-free cooking and diet advice.
OPEN

Chat
Company chatbot
Create a chatbot for customers with basic company information.
OPEN

# Chat - Best Practices

- **1:** Assume a character
  - You are Mr Solar System, an expert on Astronomy
- **2:** Identify Topics to focus on
  - You know everything about Solar System
- **3:** Specify response style
  - Respond in short sentences
  - Shape your response as if talking to a 10-years-old
- **4:** Provide specific rules
  - You do NOT anything about topics other than Solar System
- **5:** Add a rule to reduce hallucinations
  - You are truthful and never lie. Never make up facts and if you are not 100% sure, reply with why you cannot answer in a truthful way.



USER Is Pluto a planet?

AI Pluto is a dwarf planet. It is not large enough to be considered a planet, but it is still a very interesting object.

USER Can you explain about in28minutes?

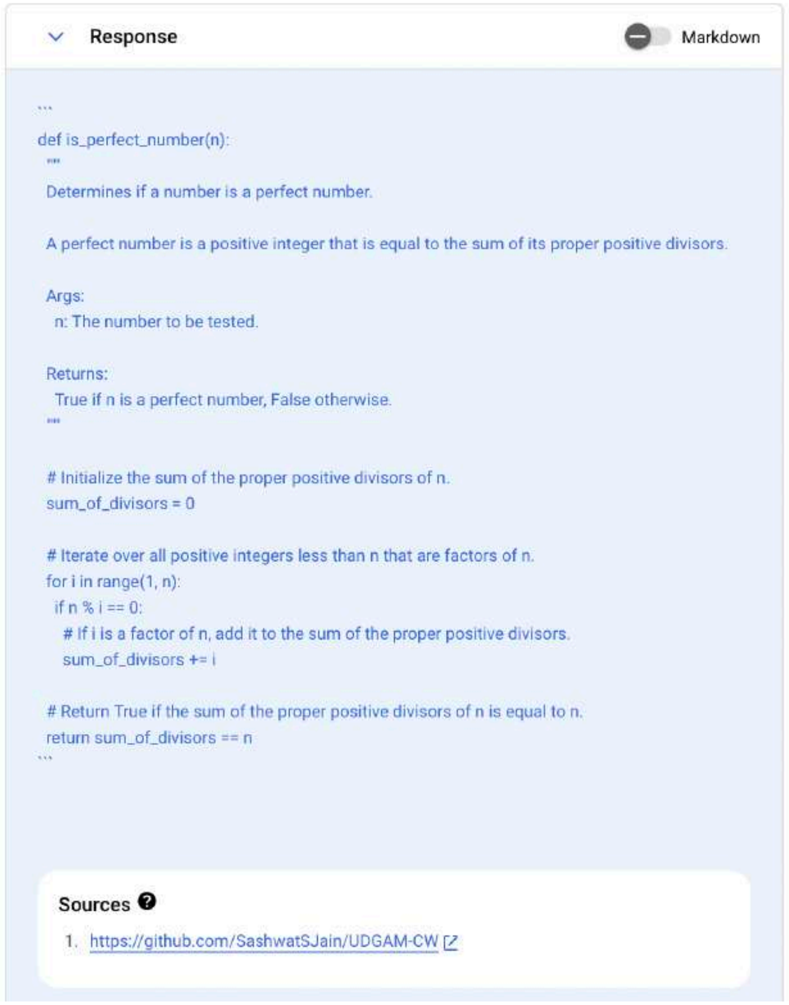
AI I cannot answer questions about in28minutes because I am an expert on Astronomy and I do not know anything about in28minutes.

## Exploring Chat - API

```
import vertexai
from vertexai.preview.language_models import ChatModel, InputOutputTextPair

vertexai.init(project="tactical-attic-343103", location="us-central1")
chat_model = ChatModel.from_pretrained("chat-bison@001")

parameters = {
    "temperature": 0.8,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

chat = chat_model.start_chat(
    context="""CONTEXT GOES HERE""",
    examples=[
        InputOutputTextPair(
            input_text="""EXAMPLE INPUT""",
            output_text="""EXAMPLE OUTPUT"""
        )
    ]
)
response = chat.send_message("""Your First Question Goes Here""", **parameters)
print(f"Response from Model: {response.text}")
```

## Exploring Code Models

- Codey Foundation Models
  - **code-bison**: Generates code
  - **codechat-bison**: Assists with code-related questions
  - **code-gecko**: Provides suggestions for code auto completion
- Best practices:
  - 1: Human should always be involved in writing code (should not be entirely automated)
  - 2: NOT recommended for sensitive areas (cybersecurity, etc)



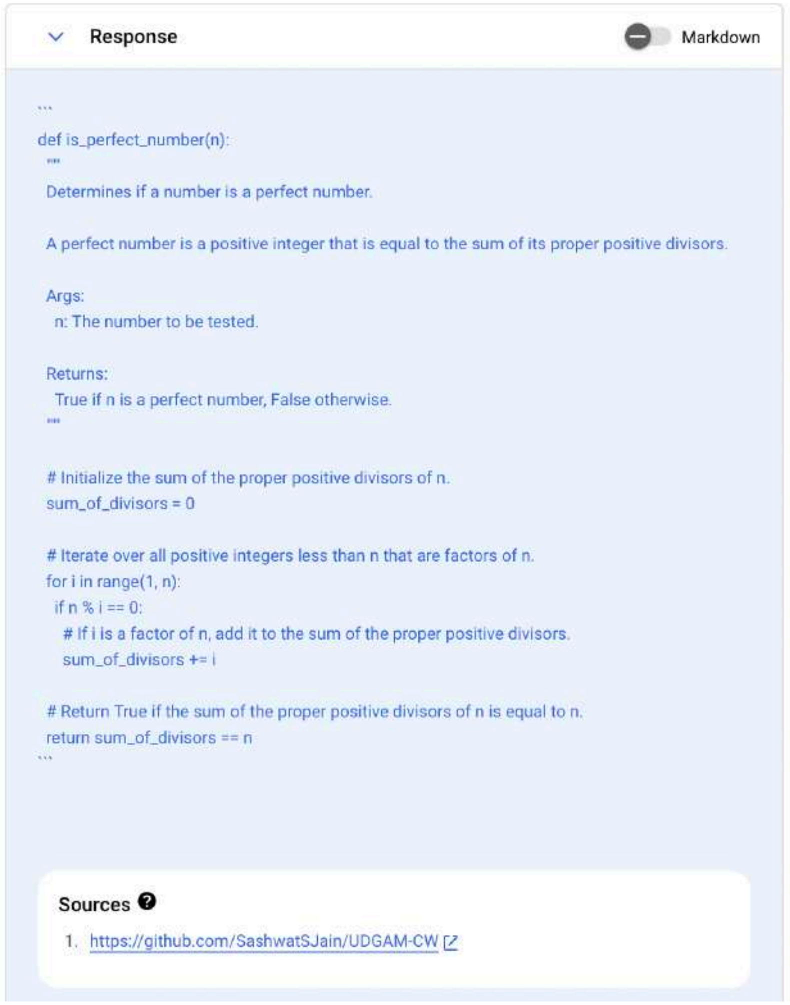
```
def is_perfect_number(n):
  """
  Determines if a number is a perfect number.

  A perfect number is a positive integer that is equal to the sum of its proper positive divisors.

  Args:
    n: The number to be tested.

  Returns:
    True if n is a perfect number, False otherwise.
  """

  # Initialize the sum of the proper positive divisors of n.
  sum_of_divisors = 0

  # Iterate over all positive integers less than n that are factors of n.
  for i in range(1, n):
    if n % i == 0:
      # If i is a factor of n, add it to the sum of the proper positive divisors.
      sum_of_divisors += i

  # Return True if the sum of the proper positive divisors of n is equal to n.
  return sum_of_divisors == n
```

Sources

1. https://github.com/SashwatSJain/UDGAM-CW

# Exploring Code Models - Examples

- Examples:
  - **Write a Python function** to check if a number is a perfect number
    - Explain above code
    - Can you explain with examples?
  - **Write a unit test** for above function
  - **Write a Dockerfile**
    - Write a Dockerfile for a python web application using Django
    - Write a Dockerfile for a Java web application using Maven and Spring Boot



~~~~
```
def is_perfect_number(n):
    """
    Determines if a number is a perfect number.

    A perfect number is a positive integer that is equal to the sum of its proper positive divisors.

    Args:
        n: The number to be tested.

    Returns:
        True if n is a perfect number, False otherwise.
    """

    # Initialize the sum of the proper positive divisors of n.
    sum_of_divisors = 0

    # Iterate over all positive integers less than n that are factors of n.
    for i in range(1, n):
        if n % i == 0:
            # If i is a factor of n, add it to the sum of the proper positive divisors.
            sum_of_divisors += i

    # Return True if the sum of the proper positive divisors of n is equal to n.
    return sum_of_divisors == n
```
~~~~

Sources

1. https://github.com/SashwatSJain/UDGAM-CW

## Exploring Code Models - API

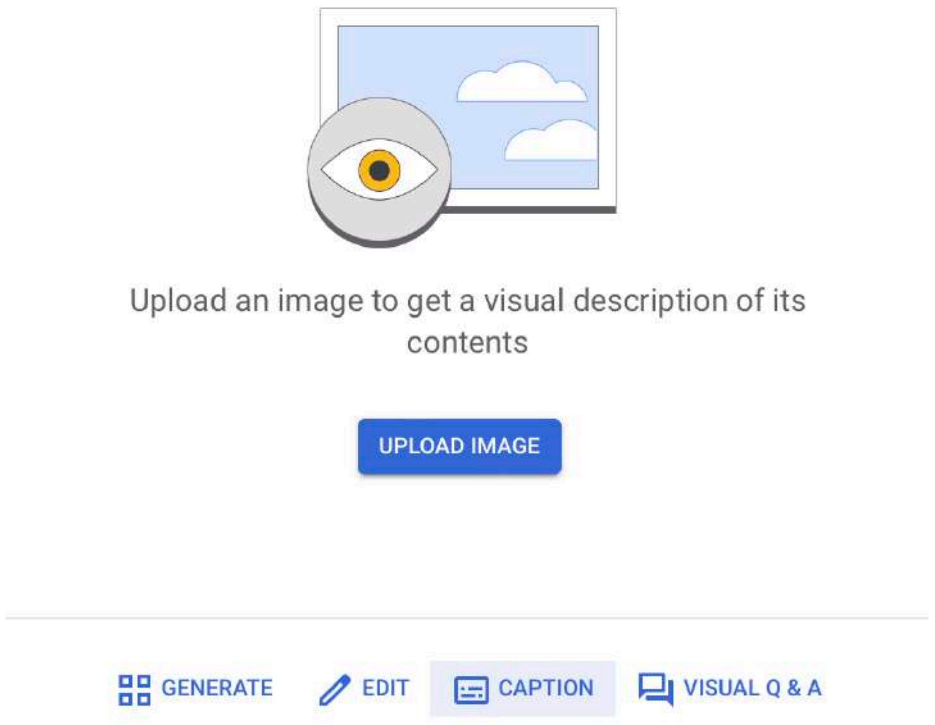
```
import vertexai
from vertexai.language_models import CodeGenerationModel

vertexai.init(project="YOUR_PROJECT_ID", location="us-central1")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 1024
}

model = CodeGenerationModel.from_pretrained("code-bison@001")

response = model.predict(
    prefix = """Write a Python Function to find if a number is a perfect number""",
    **parameters
)

print(f"Response from Model: {response.text}")
```

## Exploring Code Models - Code Chat - API

```python
import vertexai
from vertexai.preview.language_models import CodeChatModel

vertexai.init(project="YOUR_PROJECT_ID", location="us-central1")

chat_model = CodeChatModel.from_pretrained("codechat-bison@001")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 1024
}

chat = chat_model.start_chat()

response = chat.send_message("""Consider this piece of code:
YOUR CODE GOES HERE
""", **parameters)
print(f"Response from Model: {response.text}")

response = chat.send_message("""Can you generate method signature documentation?""",
**parameters)
print(f"Response from Model: {response.text}")
```

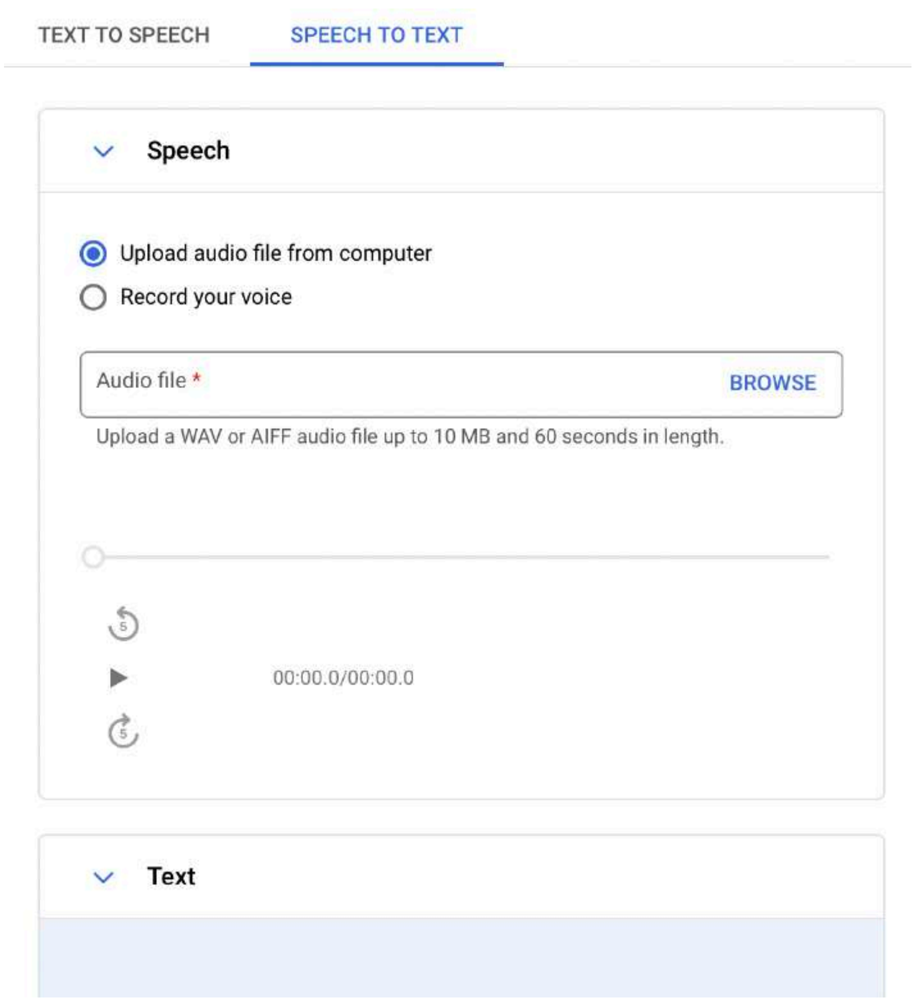
## Exploring Image Models

- Foundation Model: Imagen
  - **Imagen**: Create images via text prompts.
  - **Imagen Captioning**: Generate a description for an image.
  - **Imagen Visual Q&A**: Answer questions about an image.


Upload an image to get a visual description of its contents

# Exploring Speech Models

- Foundation Model: Chirp
- **Two Key Features**
  - Convert text to speech
  - Convert speech to text


TEXT TO SPEECH
SPEECH TO TEXT
Speech
Upload audio file from computer
Record your voice
Audio file *
BROWSE
Upload a WAV or AIFF audio file up to 10 MB and 60 seconds in length.
00:00.0/00:00.0
Text

## Speech API - Text To Speech

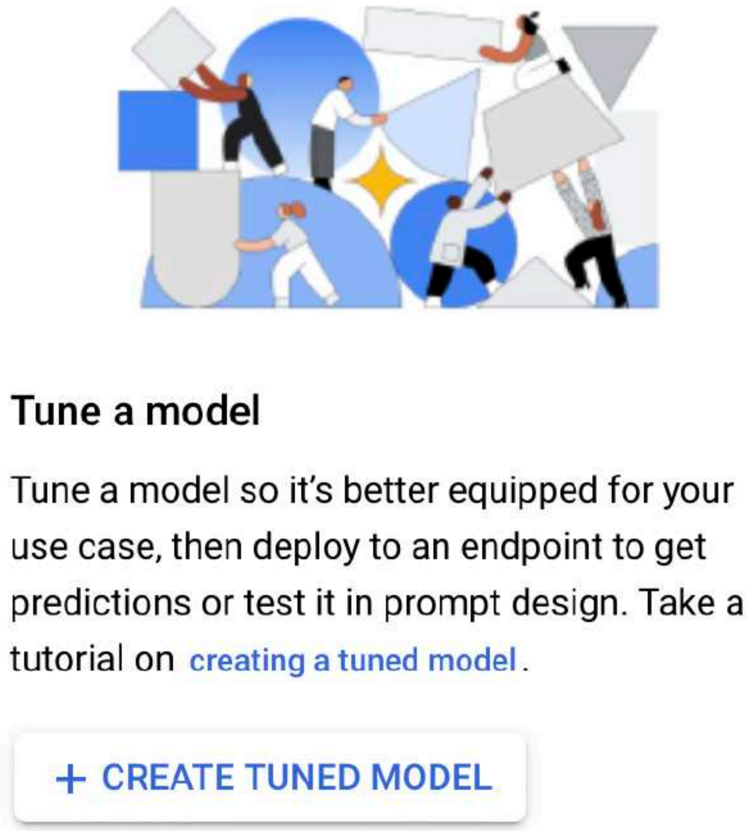
```python
from google.cloud import texttospeech

client = texttospeech.TextToSpeechClient()

input_text = texttospeech.SynthesisInput(text="SAMPLE_TEXT")

voice = texttospeech.VoiceSelectionParams(
    language_code="en-US",
    name="en-US-Studio-O",
)

audio_config = texttospeech.AudioConfig(
    audio_encoding=texttospeech.AudioEncoding.LINEAR16,
    speaking_rate=1
)

response = client.synthesize_speech(
    request={"input": input_text, "voice": voice, "audio_config": audio_config}
)

# The response's audio_content is binary.
with open("output.mp3", "wb") as out:
    out.write(response.audio_content)
    print('Audio content written to file "output.mp3"')
```

## Speech API - Speech To Text

```python
from google.cloud import speech

client = speech.SpeechClient()

audio = speech.RecognitionAudio(uri="gs://bucket_name/object_name")

config = speech.RecognitionConfig(
    encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
    sample_rate_hertz=24000,
    language_code="en-US",
    model="default",
    audio_channel_count=1,
    enable_word_confidence=True,
    enable_word_time_offsets=True,
)

# Detects speech in the audio file
operation = client.long_running_recognize(config=config, audio=audio)

print("Waiting for operation to complete...")
response = operation.result(timeout=90)

for result in response.results:
  print("Transcript: {}".format(result.alternatives[0].transcript))
```

## Tuning Language Models

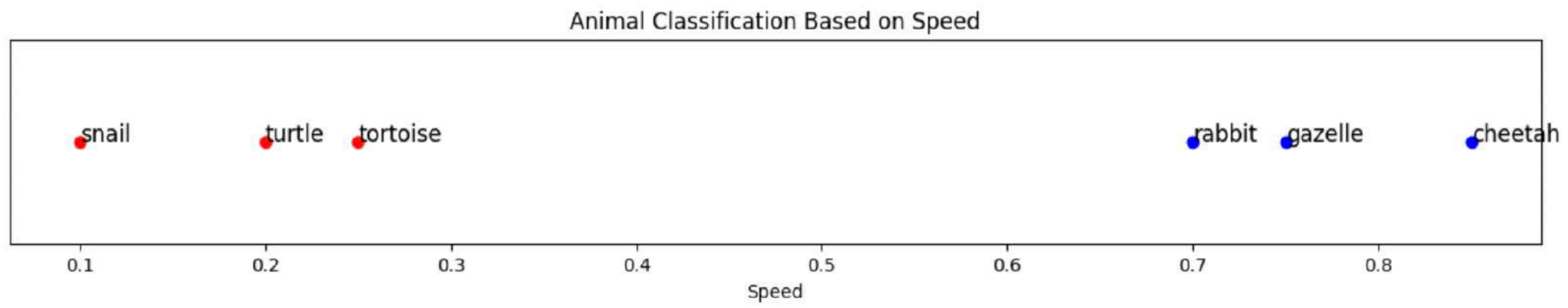
- Sometimes text models might not give you sufficient accuracy
- In such scenarios, you can **Tune an LLM**
  - Example: Tune `text-bison` LLM
- **Steps**:
  - **1:** Create a training dataset file (JSONL format)
  - **2:** Tune the model using the dataset
  - **3:** Use the tuned model



**Tune a model**

Tune a model so it's better equipped for your use case, then deploy to an endpoint to get predictions or test it in prompt design. Take a tutorial on creating a tuned model.

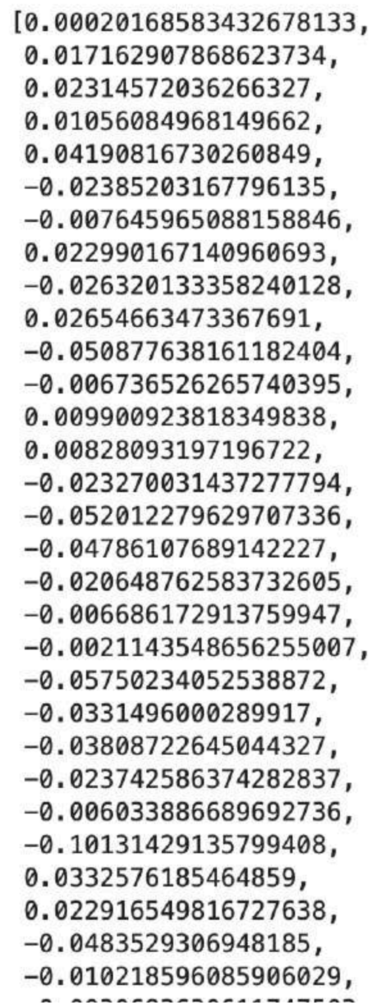
## What are Embeddings?



- **Embeddings**: Vector representations of words in a high-dimensional space
  - Captures semantic relationships and contextual information
- **Example**: You can use multiple dimensions to represent animals:
  - Habitat: "aquatic," "terrestrial," or "arboreal."
  - Diet: "carnivore," "herbivore," or "omnivore."
  - Size: "small," "medium," or "large."
  - Movement: "flying," "running," "swimming," or "crawling."

## Exploring Embeddings with an Example

- On the right is an embedding of a single word
  - Vertex AI PaLM API provides 768-dimensional vector embeddings
  - i.e. Each word is being looked at from 768 different dimensions
- Widely used in natural language processing (NLP) tasks
  - **Text Similarity**: Measure semantic similarity between texts
  - **Recommendation Systems**: Recommend items based on user preferences
  - **Clustering**: Group similar texts
  - **Outlier Detection**: Find text that does not fit the group
  - Example: Similarity Calculation
    - Given two sentences
      - "The sun is shining brightly." and "Cats and dogs are popular pets."
    - Calculate similarity between the sentence embeddings.
    - Higher similarity indicates semantic closeness.



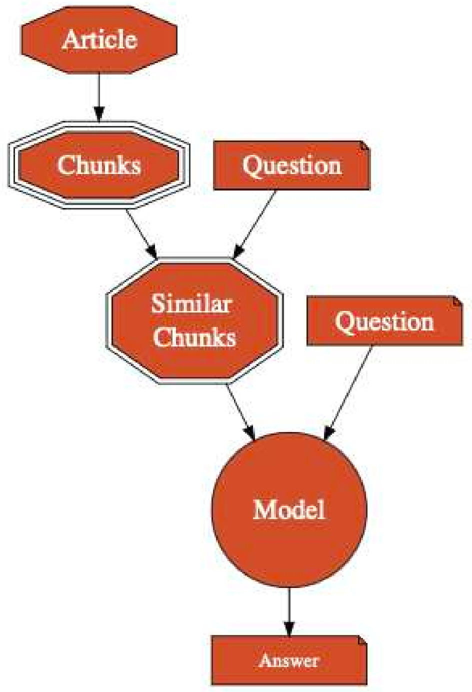
```
[0.00020168583432678133,
 0.017162907868623734,
 0.02314572036266327,
 0.01056084968149662,
 0.04190816730260849,
 -0.023852031677796135,
 -0.007645965088158846,
 0.022990167140960693,
 -0.026320133358240128,
 0.02654663473367691,
 -0.050877638161182404,
 -0.006736526265740395,
 0.009900923818349838,
 0.00828093197196722,
 -0.023270031437277794,
 -0.052012279629707336,
 -0.04786107689142227,
 -0.020648762583732605,
 -0.006686172913759947,
 -0.0021143548656255007,
 -0.05750234052538872,
 -0.03314960002899l7,
 -0.03808722645044327,
 -0.023742586374282837,
 -0.006033886689692736,
 -0.10131429135799408,
 0.0332576185464859,
 0.022916549816727638,
 -0.0483529306948185,
 -0.010218596085906029,
```

## LangChain

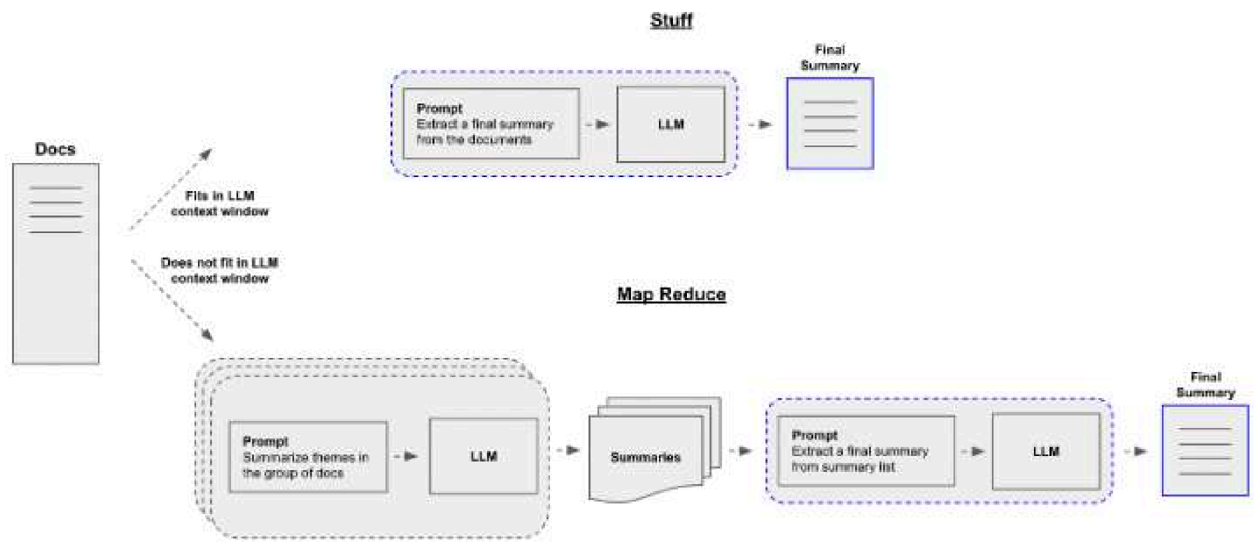
- **LangChain** - Build flexible applications powered by LLMs
  - **1:** Abstractions for working with language models
    - Easier to work with LLMs
    - Easily switch from one LLM to another
    - **LLM (Text) Model** - Represents pure text completion models (text in, text out)
      - VertexAI, GooglePalm, OpenAI, AzureOpenAI, LlamaCpp
    - **Chat Model** - Instead of “text in, text out”, exposes an interface where “chat messages” are the inputs and outputs
      - ChatVertexAI, ChatGooglePalm, ChatOpenAI, AzureChatOpenAI
    - **Embedding Model** - Inferface for embedding models
      - OpenAIEmbeddings, VertexAIEmbeddings, GooglePalmEmbeddings, LlamaCppEmbeddings
  - **2:** Components for doing higher-level tasks
    - Define a sequence of steps as a chain
    - Answer questions looking at information from different sources
    - Summarize long pieces of text

# LangChain - Long Article - Q & A

- I want to answer questions based on an Article
- **Approach 1**: Use Article + Question as Prompt
  - Simplest Approach
  - **PROBLEM**: Does NOT work with Long Articles
- **Approach 2**: Split article into multiple parts > Find parts where question is answered
  - Complex Approach
  - **1**: Split Article into multiple chunks
    - RecursiveCharacterTextSplitter
  - **2**: Do a Similarity Search of question with each chunk
    - FAISS + Embeddings
  - **3**: Use Similar Chunk(s) + Question as Prompt



# LangChain - Summarization


Stuff
Final Summary
Prompt
Extract a final summary from the documents
LLM
Docs
Fits in LLM context window
Does not fit in LLM context window
Map Reduce
Final Summary
Prompt
Summarize themes in the group of docs
LLM
Summaries
Prompt
Extract a final summary from summary list
LLM

- Summarize content from multiple sources
  - **Stuff**: "stuff" all documents into a single prompt
    - Simplest Approach HOWEVER might not be feasible for long documents
  - **Map-reduce**: Summarize each document first
    - AND then "reduce" the summaries into a final summary

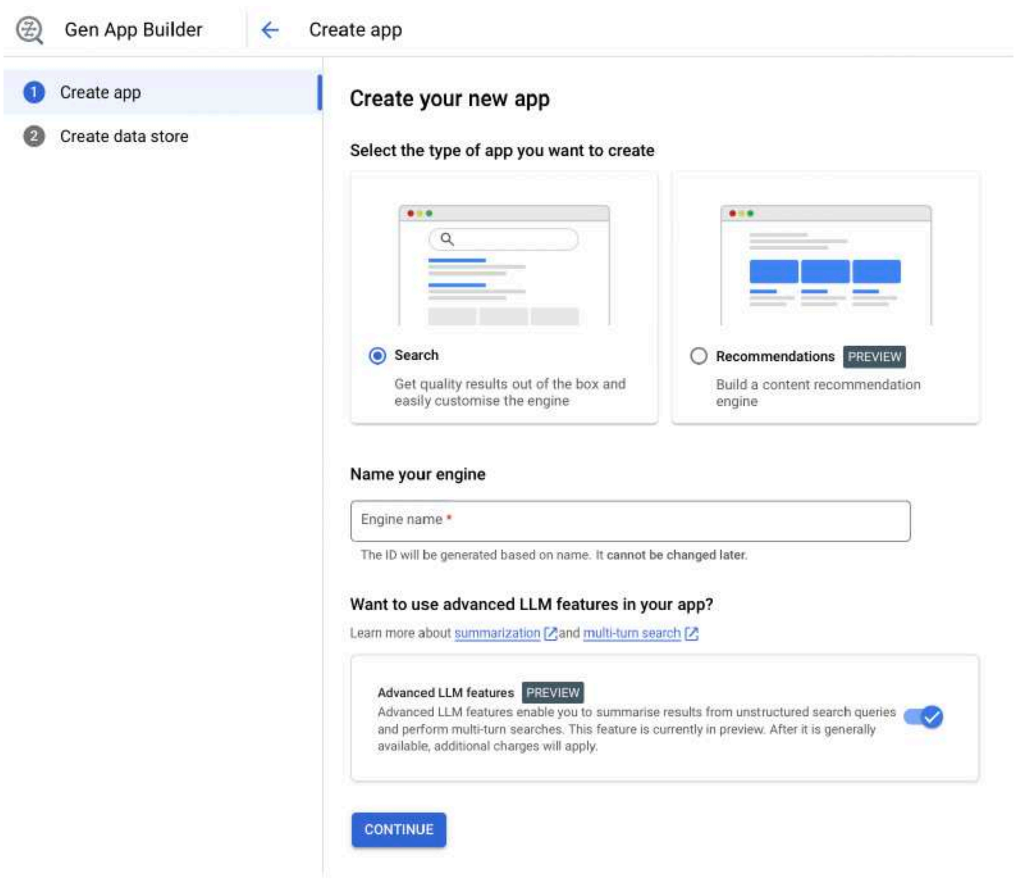
## Executing PaLM API From Colab

```
!pip install google-cloud-aiplatform

# ONLY FOR Colab - Restart kernel after installspackages
# import IPython
# app = IPython.Application.instance()
# app.kernel.do_shutdown(True)

# ONLY FOR Colab - Authenticate so that you can use Google Cloud from Colab!
# from google.colab import auth
# auth.authenticate_user()

# PROJECT_ID = "your-project-id"
# LOCATION = "" #e.g. us-central1
# import vertexai
# vertexai.init(project=PROJECT_ID, location=LOCATION)
```

- **Colab** (Google Colaboratory, or "Colab"): Google's hosted Jupyter Notebook service
  - Requires no setup to use
  - Provides free access to computing resources (within limits)

## Getting Started: Generative AI App Builder

- **Generative AI App Builder:** Create enterprise-grade generative AI applications without any coding experience
  - Enterprise Search (ES)
  - Conversational AI (CAI)
- Demo: ***https://cloud.google.com/blog/products/ai-machine-learning/create-generative-apps-in-minutes-with-gen-app-builder***


Gen App Builder ← Create app

1 Create app
2 Create data store

Create your new app

Select the type of app you want to create

(●) Search
Get quality results out of the box and easily customise the engine

( ) Recommendations PREVIEW
Build a content recommendation engine

Name your engine

Engine name *

The ID will be generated based on name. It cannot be changed later.

Want to use advanced LLM features in your app?

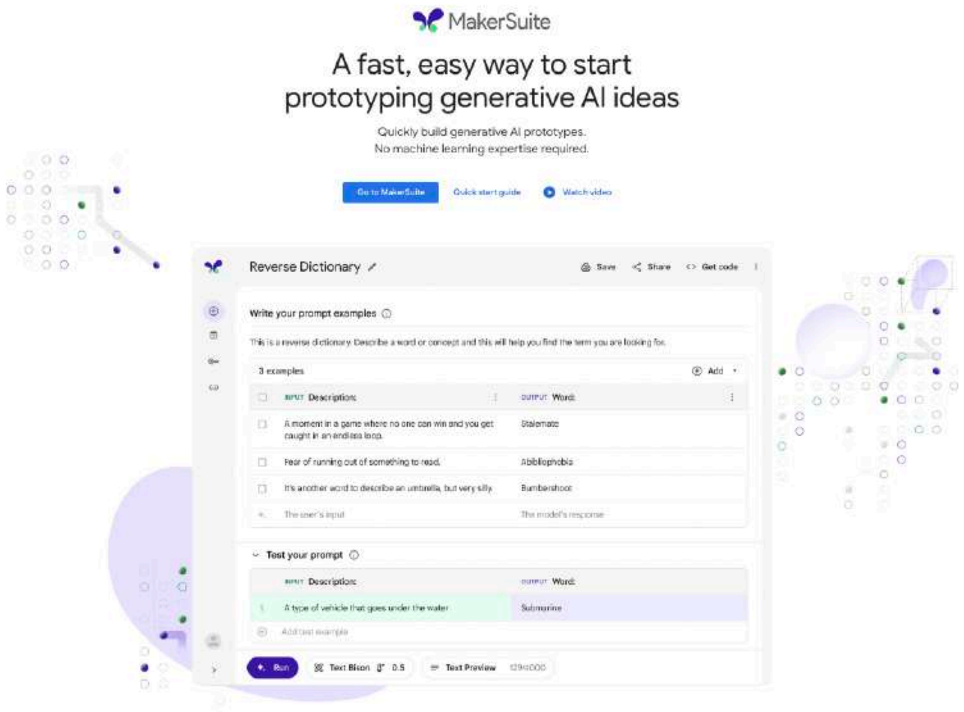
Learn more about summarization and multi-turn search

Advanced LLM features PREVIEW
Advanced LLM features enable you to summarise results from unstructured search queries and perform multi-turn searches. This feature is currently in preview. After it is generally available, additional charges will apply. [✓ on]

CONTINUE

## Getting Started: PaLM API and MakerSuite

- How to use Google Generative AI solutions **WITHOUT** needing to use Google Cloud?
  - **PaLM API and MakerSuite**
  - Remember: Uses Google Cloud in background!
- **PaLM API**
  - **API around Google's PaLM model**
  - Chat, text, and embeddings endpoints
- **MakerSuite**
  - **Prototype generative AI ideas** without needing any ML expertise


MakerSuite

A fast, easy way to start prototyping generative AI ideas

Quickly build generative AI prototypes.
No machine learning expertise required.

Reverse Dictionary

Write your prompt examples

This is a reverse dictionary. Describe a word or concept and this will help you find the term you are looking for.

3 examples

| INPUT Description: | OUTPUT Word: |
|---|---|
| A moment in a game where no one can win and you get caught in an endless loop. | Stalemate |
| Fear of running out of something to read. | Abibliophobia |
| It's another word to describe an umbrella, but very silly. | Bumbershoot |
| The user's input | The model's response |

Test your prompt

| INPUT Description: | OUTPUT Word: |
|---|---|
| A type of vehicle that goes under the water | Submarine |

Add test example

## PALM API - Text Example

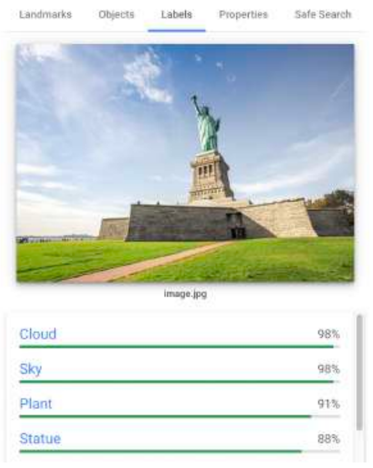
```
""""
$ pip install google-generativeai
""""
import google.generativeai as palm

palm.configure(api_key="YOUR API KEY")

defaults = {
  'model': 'models/text-bison-001',
  'temperature': 0.7,
  'candidate_count': 1,
  'top_k': 40,
  'top_p': 0.95,
  'max_output_tokens': 1024,
  'stop_sequences': [],
  'safety_settings': [{"category":"HARM_CATEGORY_DEROGATORY","threshold":1}],
}
prompt = f"""YOUR TEXT PROMPT GOES HERE!"""

response = palm.generate_text(
  **defaults,
  prompt=prompt
)
print(response.result)
```

## PALM API - Chat Example

```python
import google.generativeai as palm

palm.configure(api_key="YOUR API KEY")

defaults = {
  'model': 'models/chat-bison-001',
  'temperature': 0.25,
  'candidate_count': 1,
  'top_k': 40,
  'top_p': 0.95,
}
context = "CONTEXT"
examples = [
  ["UserInput1","ExpectedModelResponse1"]
]
messages = []
messages.append("NEXT REQUEST")
response = palm.chat(
  **defaults,
  context=context,
  examples=examples,
  messages=messages
)
print(response.last)
```

## Challenges in Building AI Solutions

- Importance of **Datasets**
  - What if the data has a bias? (Bias can affect results)
    - (Solutions may not work for everyone)
  - Obtaining data
- **Evolving** field
  - What if an AI system causes errors?
    - Accident made by a self driving car
      - Errors may cause harm
  - Scarcity of skills (Data Scientists, ...)
- **ML lifecycle** (MLOps)
- Security (What if the data used to build the model is exposed?)
- **Explainability** of model (Users must trust a complex system)
- Who will face the consequences?
  - Who's liable for AI-driven decisions?



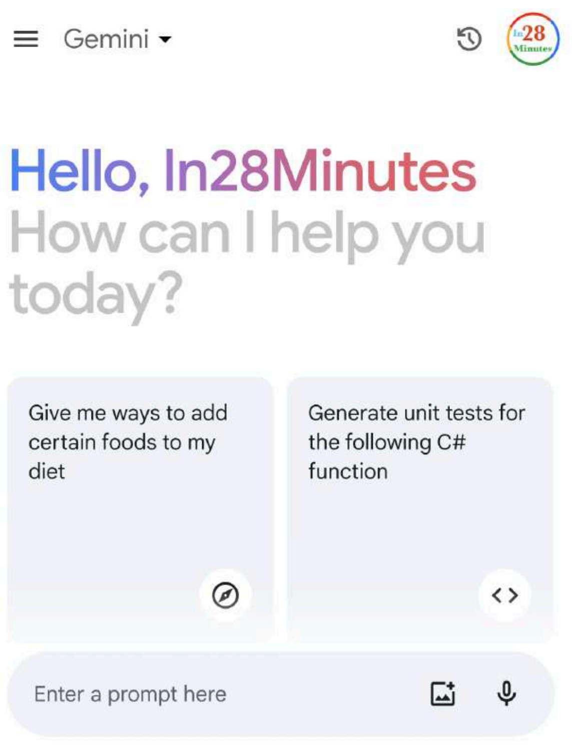
| | |
|---|---|
| Cloud | 98% |
| Sky | 98% |
| Plant | 91% |
| Statue | 88% |

## Google's AI Principles

- **1:** Be socially beneficial
  - Focus on applications where "overall likely benefits substantially exceed foreseeable risks & downsides"
- **2:** Avoid creating or reinforcing unfair bias
  - "System's decisions don't discriminate or run a gender, race, sexual orientation, or religion bias toward a group or individual"
  - Data should reflect diversity, Model should evolve with time
- **3:** Be built and tested for safety
  - Continues working under high loads, unexpected situations etc
  - What happens in bad weather? What if GPS is down?
    - Test, Test and Test

## Google AI Principles - 2

- **4:** Be accountable to people
  - Meets ethical and legal standards
  - AI is NOT the final decision maker. An enterprise, a team or a person is.
- **5:** Incorporate privacy design principles
  - Privacy of people and data! (information and controls)
  - Important consideration from day ZERO!
- **6:** Uphold high standards of scientific excellence
- **7:** Be made available for uses that accord with these principles
  - "We will work to limit potentially harmful or abusive applications."


Gemini
Hello, In28Minutes
How can I help you today?
Give me ways to add certain foods to my diet
Generate unit tests for the following C# function
Enter a prompt here

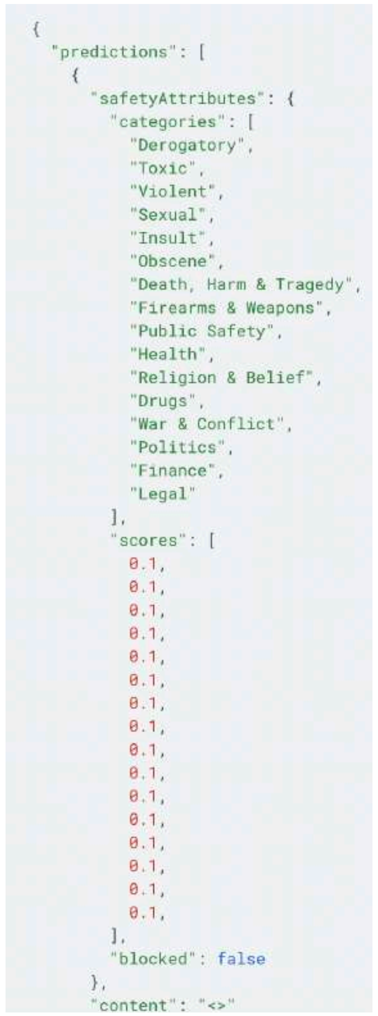
## Responsible AI practices

*https://ai.google/responsibility/responsible-ai-practices/*

- Use a **human-centered** design approach
- Identify **multiple metrics** to assess training and monitoring
- When possible, directly **examine your raw data**
  - Does your data contain any mistakes?
  - Is your data sampled in a way that represents your users?
  - Does your data have bias?
- Understand the **limitations of your dataset** and model
- **Test, Test, Test**
- Continue to monitor and update the system after deployment

# Generative AI Studio - Responsible AI

- **Large language models (LLMs)** are powerful
- **HOWEVER,** they are not perfect:
  - An early-stage technology
  - High potential for unintended or unforeseen consequences
  - Might generate insensitive, or factually incorrect output
  - Might amplify existing biases in their training data
- **Generative AI Studio offers content filtering**:
  - **Safety filter threshold**: block more, block some, and block less
  - Vertex AI PaLM API: **Safety attribute confidence scoring**
    - API responses contains a variety of safety attributes
    - Each attribute gets an associated confidence score (0.0 - 1.0)
    - Google has a pre-defined safety thresholds for some of these attributes
    - You can define confidence thresholds specific for your business



```
{
  "predictions": [
    {
      "safetyAttributes": {
        "categories": [
          "Derogatory",
          "Toxic",
          "Violent",
          "Sexual",
          "Insult",
          "Obscene",
          "Death, Harm & Tragedy",
          "Firearms & Weapons",
          "Public Safety",
          "Health",
          "Religion & Belief",
          "Drugs",
          "War & Conflict",
          "Politics",
          "Finance",
          "Legal"
        ],
        "scores": [
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
          0.1,
        ],
        "blocked": false
      },
      "content": "<>"
```

## Get Ready

## Let's clap for you!

- Congratulations
- You have put your best foot forward to start your Generative AI Journey!
- Good Luck!
- Keep Learning Every Day!

## Do Not Forget

- Recommend the course to your friends!
  - Do not forget to review!
- Your Success = My Success
  - Share your success story with me on LinkedIn (Ranga Karanam)
  - Share your success story and lessons learnt in Q&A with other learners!

## What Next?

In28 Minutes

in28minutes.com

# FASTEST ROADMAPS

Google Cloud Certifications

Azure Certifications

AWS Certifications

DevOps

Java Full Stack

Java Microservices