

Coding the Future of AI

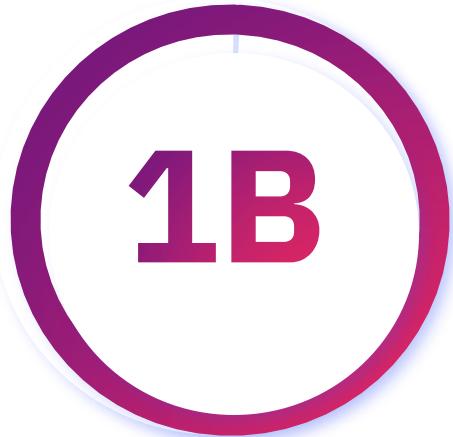




**Every code
program**
**will be reinvented
with AI**



New code will be
built that weren't
possible before



New code applications
built over the next 24-
36 months



Organizations
accelerating code
modernization to bring
AI into existing apps



Developers **using AI** in
their toolset

The reality

14%

**Conversion rate from
AI PoC to production**

74%

**Organizations in GenAI
planning/
experimentation stage**

Factors hindering GenAI production

41% Lack of developer skills and tools

38% Excessive costs outstripping ROI goals

36% Infrastructure performance and availability

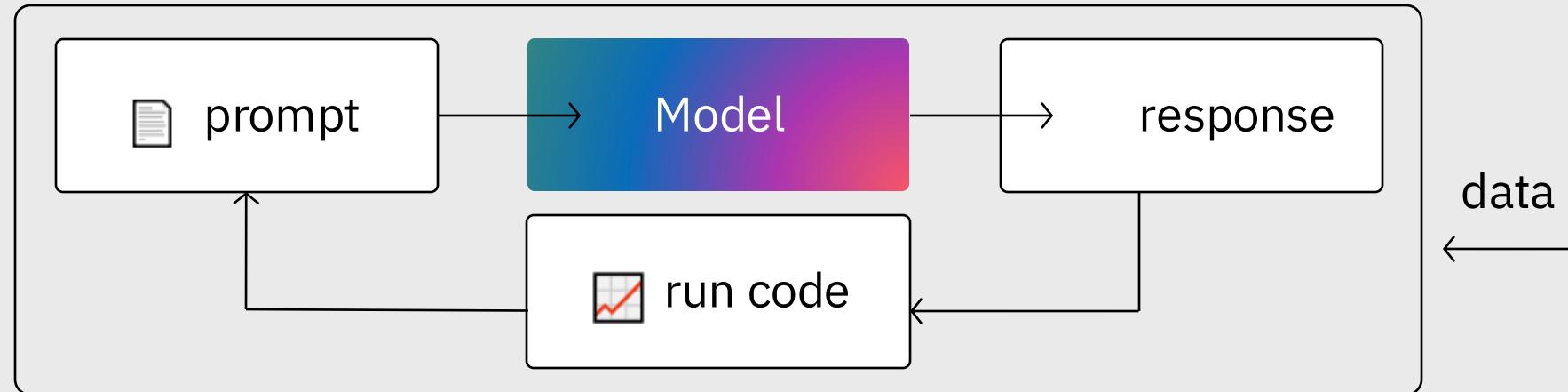
35% Poor IT and business coordination

De-mystifying AI Development

At a basic level

Providing prompt to a model

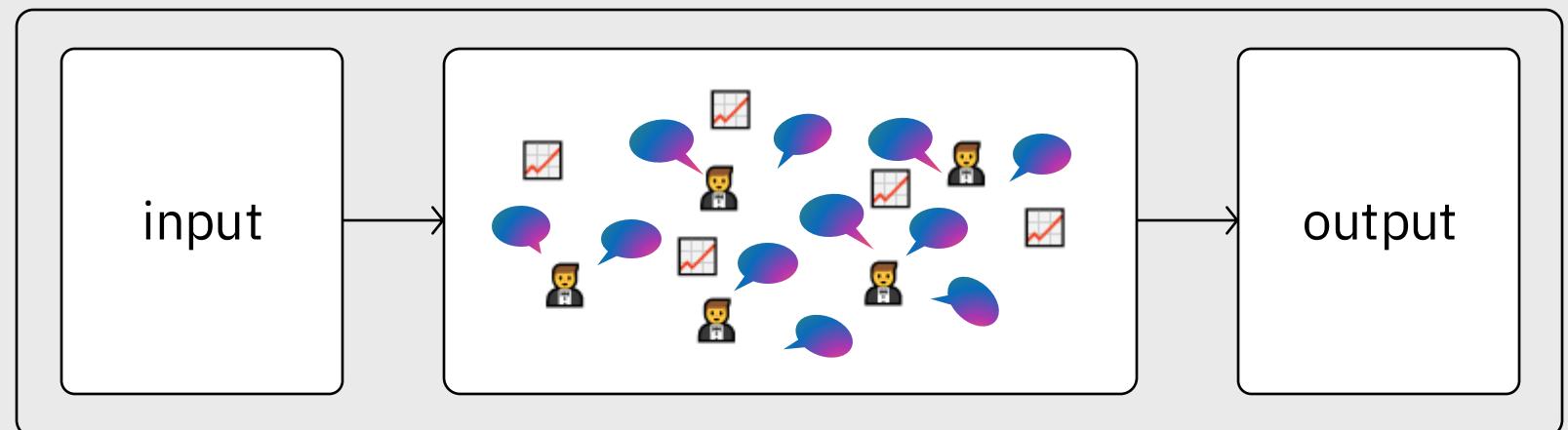
Acting on the response



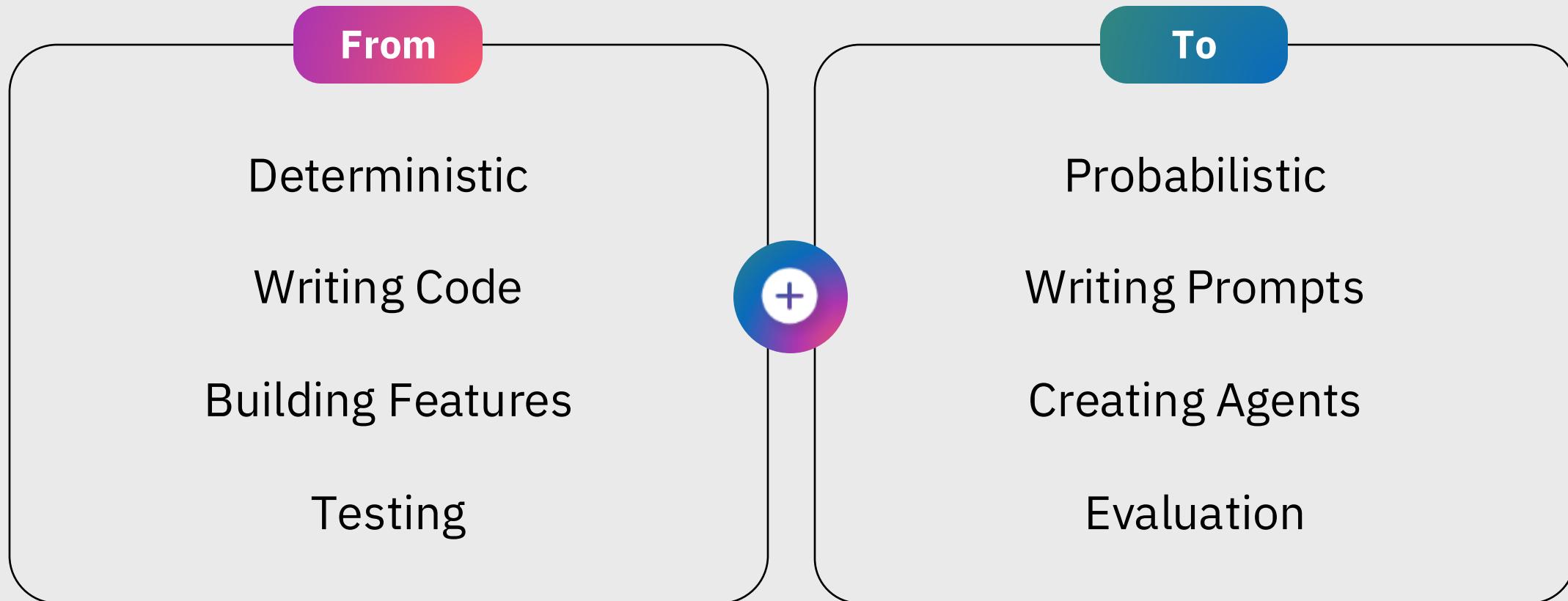
At scale

Holding many different conversations with models & agents

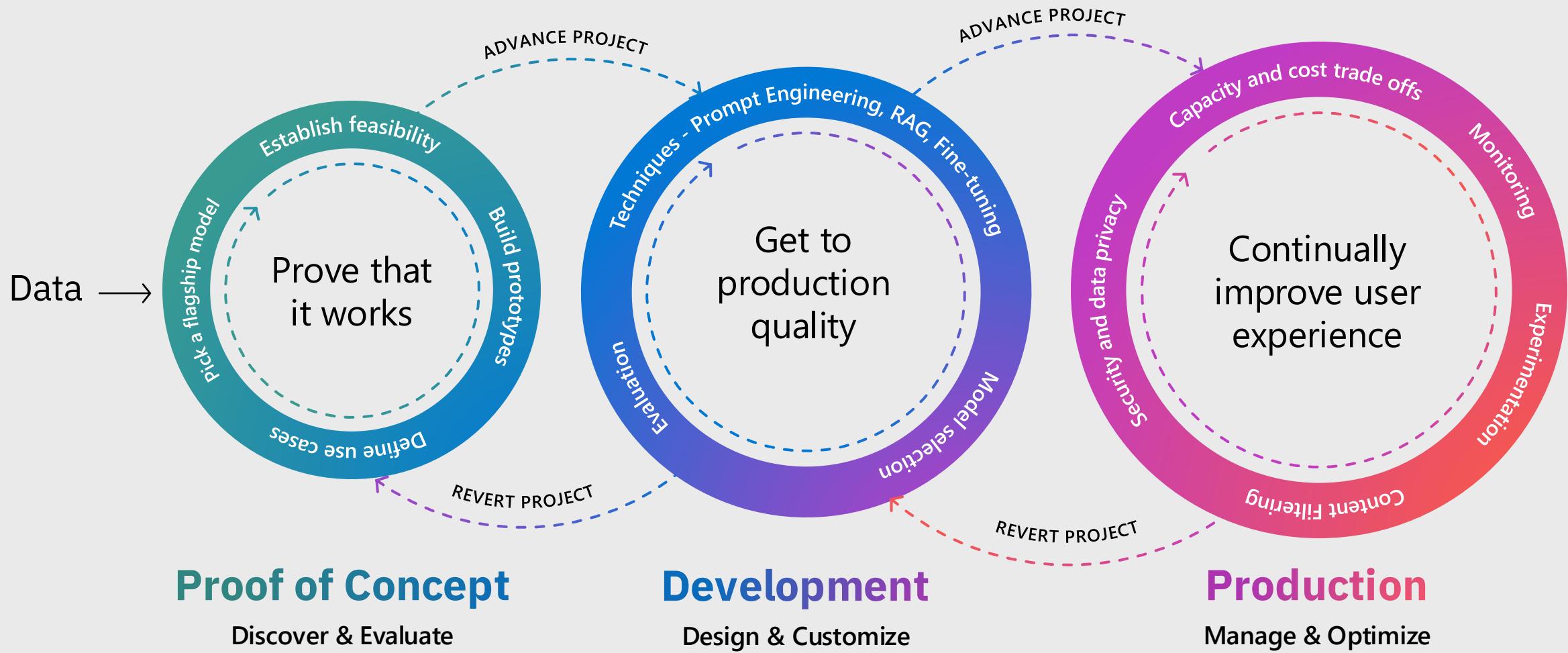
Extracting information into useful results



The way we write code is changing

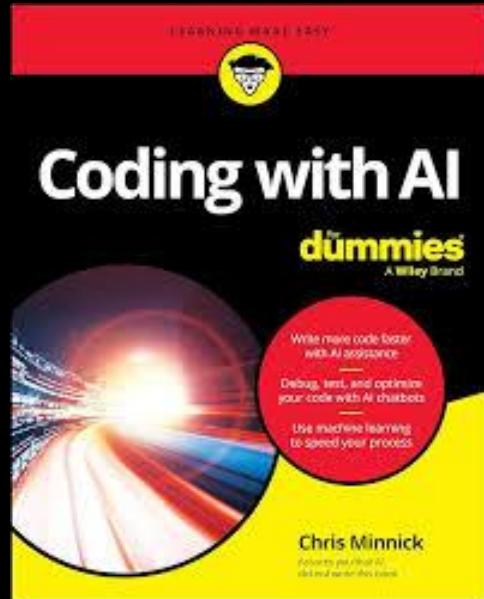


Developers start small and iterate



Leveraging AI in Coding

Techniques, Tools, and Best Practices



I. Techniques & Technologies

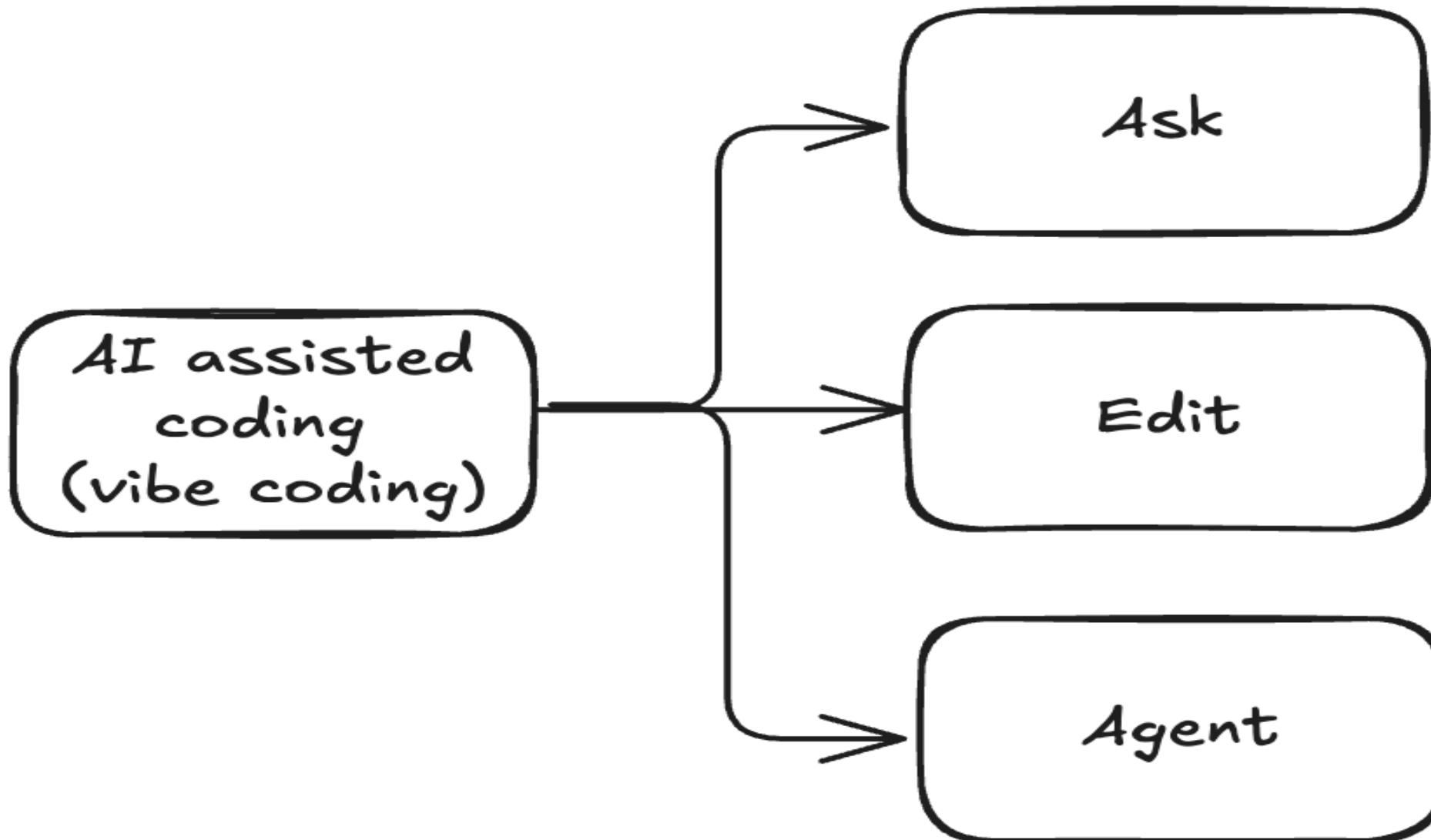
1. Benefits
2. Parsing ML/DL
3. AI coding tools
4. Coding w chatbots

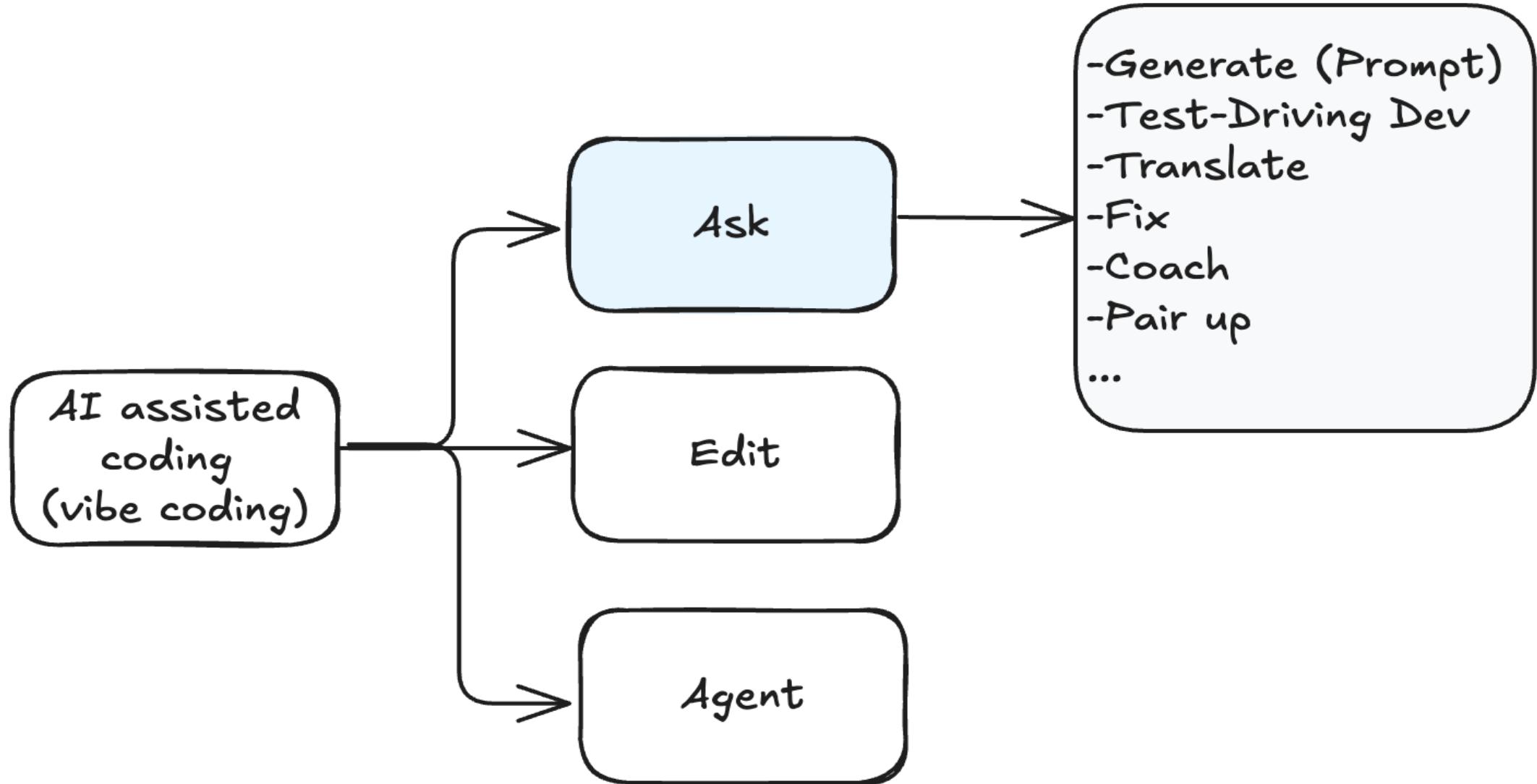
II. Using AI to write code

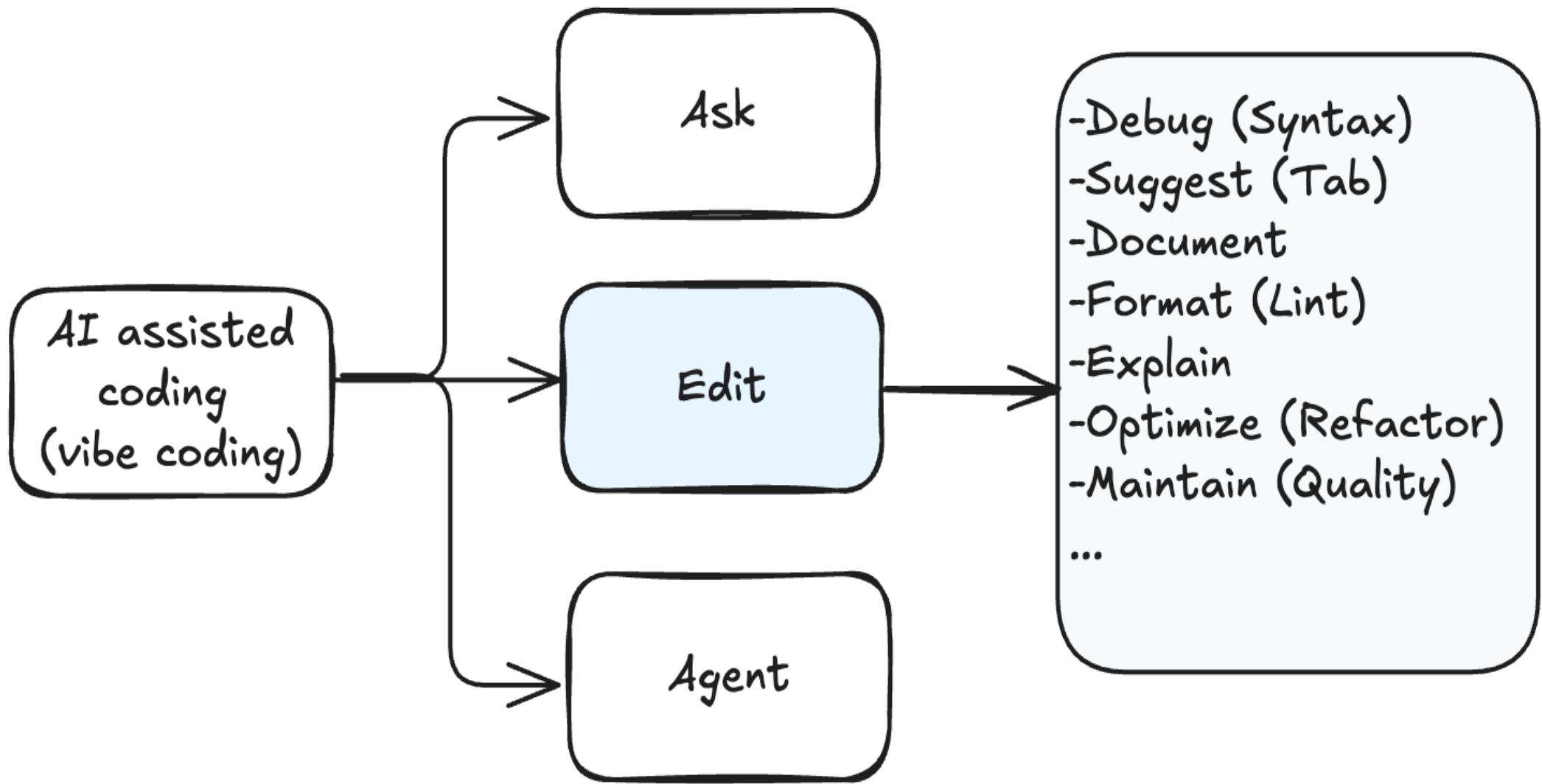
5. From plan to prototype
6. Formatting & improving code
7. Find, eliminate bugs
8. Translate & optimize code

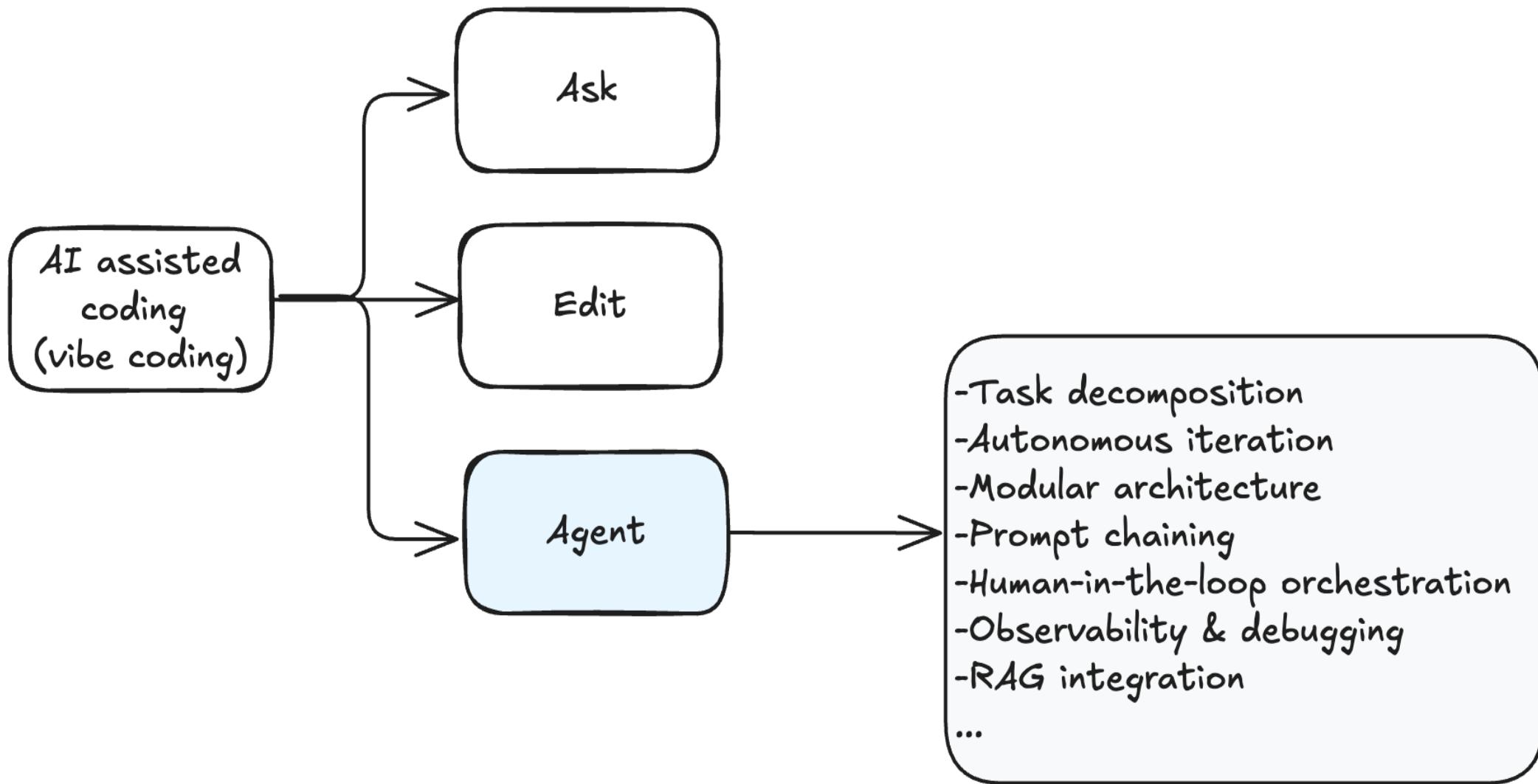
III. Test, Document & Maintain code

9. Testing your code
10. Documenting your code
11. Maintaining your code









Risk mitigation plan for AI-assisted Coding in Education

Key Concerns

Blind trust

Prevent shortcut learning

Demystify the tool

Safeguards

Verify AI code & understand it before use

Use manual coding & AI to build foundational skills

Teach how AI coding tools work



Risk mitigation plan for AI-assisted Coding in Education

Key Concerns

Skills gap



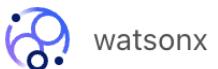
No human-in-the loop

Safeguards

Foster critical thinking & peer reviews to reduce overconfidence in unverified outputs

Mandate code reviews & discussions on AI-generated code

Code generation

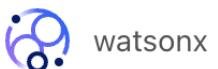


Hi, how can I help you today?

You

Write a Python script that connects to a MongoDB database, retrieves documents from a 'users' collection, and prints usernames and their corresponding email addresses.

:



Assisted by watsonx Code Assistant
import pymongo

Connect to MongoDB
client = pymongo.MongoClient("mongodb://localhost:27017/")

Access the 'users' collection
db = client["mydatabase"]
collection = db["users"]

Retrieve documents and print usernames and email addresses
for document in collection.find():
 print(f"Username: {document['username']}, Email: {document['email']}")

□

Note: This script assumes that a MongoDB database named 'mydatabase' exists on the local machine, and a 'users' collection with documents containing 'username' and 'email' fields.



Code completion

primes.py 1 ● main.py 2 Settings

Users > jportilla > Downloads > primes.py > ...

```
1 #calculate primes less 100
2 #172.27.36.41:11434
3
4 Explain | Document | Unit Test
5 def calculatePrimes(n):
    primes = []
        for num in range(2, n+1):
            is_prime = True
            for i in range(2, int(num**0.5) + 1):
                if num % i == 0:
                    is_prime = False
                    break
            if is_prime:
                primes.append(num)
    return primes
```

Code Explanation

Chat session 1

watsonx

Hi, how can I help you today?

You

main.py explain this code :

watsonx

This code is a Python script that uses the MLflow library to train a machine learning model. The script takes in several input arguments, including the path to the input data, the test-train ratio, the number of estimators, the learning rate, and the name of the registered model. The script starts by parsing the input arguments using the argparse module. It then starts a new MLflow run and enables autologging for the sklearn module.

Next, the script prepares the data by reading an Excel file using the pandas library, splitting the data into training and testing sets, and extracting the label column.

The script then trains a GradientBoostingClassifier model using the training data and logs the number of samples and features. It also logs the classification report for the predicted values.

Finally, the script registers the model with MLflow and saves it to a file.

Overall, this code is a good example of how to use MLflow to train and register a machine learning model in Python.

Upvote

src > main.py > ...

```
10 ✓ def main():
11     """Main function of the script."""
12
13     # input and output arguments
14     parser = argparse.ArgumentParser()
15     parser.add_argument("--data", type=str, help="path to input data")
16     parser.add_argument("--test_train_ratio", type=float, required=False, default=0.25)
17     parser.add_argument("--n_estimators", required=False, default=100, type=int)
18     parser.add_argument("--learning_rate", required=False, default=0.1, type=float)
19     parser.add_argument("--registered_model_name", type=str, help="model name")
20     args = parser.parse_args()
21
22     # Start Logging
23     mlflow.start_run()
24
25     # enable autologging
26     mlflow.sklearn.autolog()
27
28 #####
29 #<prepare the data>
30 #####
31 print(" ".join(f'{k}={v}' for k, v in vars(args).items()))
32
33 print("input data:", args.data)
34
35 credit_df = pd.read_excel(args.data, header=1, index_col=0)
36
37 mlflow.log_metric("num_samples", credit_df.shape[0])
38 mlflow.log_metric("num_features", credit_df.shape[1] - 1)
39
40 ✓ train_df, test_df = train_test_split(
41     credit_df,
42     test_size=args.test_train_ratio,
43 )
44 #####
45 #</prepare the data>
```

Code Translation

Chat session 1

AI

write similar function in java main.py :

watsonx

```
// Assisted by watsonx Code Assistant
//main.py
import os
import argparse
import pandas as pd
import mlflow
import mlflow.sklearn
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

def main():
    """Main function of the script."""

    // input and output arguments
    parser = argparse.ArgumentParser()
    parser.add_argument("--data", type=str, help="path to input data")
    parser.add_argument("--test_train_ratio", type=float, required=False, default=0.25)
    parser.add_argument("--n_estimators", required=False, default=100, type=int)
    parser.add_argument("--learning_rate", required=False, default=0.1, type=float)
    parser.add_argument("--registered_model_name", type=str, help="model name")
    args = parser.parse_args()

    // Start Logging
    mlflow.start_run()

    // enable autologging
    mlflow.sklearn.autolog()

#####
#<prepare the data>
#####
print(" ".join(f"{k}={v}" for k, v in vars(args).items()))

print("input data:", args.data)

credit_df = pd.read_excel(args.data, header=1, index_col=0)

mlflow.log_metric("num_samples", credit_df.shape[0])
mlflow.log_metric("num_features", credit_df.shape[1] - 1)

train_df, test_df = train_test_split(
    credit_df,
    test_size=args.test_train_ratio,
)
#####
#</prepare the data>
#####

#####
#<train the model>
#####
# Extracting the label column
y_train = train_df.pop("default payment next month")

# convert the dataframe values to array
X_train = train_df.values

# Extracting the label column
y_test = test_df.pop("default payment next month")

# convert the dataframe values to array
X_test = test_df.values
```

Enter / for commands

Code Test

Chat session 1

Prompt library

- Develop a GoLang HTTP server that serves a simple REST API for managing a collection of books. Implement CRUD operations (Create, Read, Update, Delete) for book entities, storing data in an in-memory data structure.
- Develop JUnit test cases for a Java class that represents a simple calculator with methods for addition, subtraction, multiplication, and division.
- Write Golang test functions for a package that handles image processing. Include test cases for image resizing and format conversion.
- Write JUnit test cases for a Java class that implements a stack data structure. Include tests for push, pop, and peek operations, as well as handling edge cases such as an empty stack.
- Add comments to a C++ class representing a file handling utility. Include comments for methods handling file reading, writing, and error handling.

watsonx

Here's an example of JUnit test cases for a Java class that implements a stack data structure:

```
// Assisted by watsonx Code Assistant
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class StackTest {

    Stack<Integer> stack = new Stack<>();

    @Test
    public void testPush() {
        stack.push(1);
        assertEquals(1, stack.peek());
        stack.push(2);
        assertEquals(2, stack.peek());
    }

    @Test
    public void testPop() {
        stack.push(1);
        stack.push(2);
        assertEquals(2, stack.pop());
        assertEquals(1, stack.pop());
    }

    @Test
    public void testPeek() {
        stack.push(1);
        assertEquals(1, stack.peek());
        stack.push(2);
        assertEquals(2, stack.peek());
    }

    @Test
    public void testEmptyStack() {
        Exception exception = assertThrows(Exception.class, () -> stack.peek());
        assertEquals("Stack is empty", exception.getMessage());
    }
}
```

Code Translation

/translate from python to java main.py

:



watsonx

```
// Assisted by watsonx Code Assistant
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
import java.util.stream.Collectors;

public class Main {
    public static void main(String[] args) {
        String data = args[0];
        double testTrainRatio = Double.parseDouble(args[1]);
        int nEstimators = Integer.parseInt(args[2]);
        double learningRate = Double.parseDouble(args[3]);
        String registeredModelName = args[4];

        // Start Logging
        Mlflow.startRun();

        // enable autologging
        Mlflow.sklearn.autolog();

        // Prepare the data
        System.out.println("input data: " + data);
        List<String> lines = null;
        try {
            lines = Files.readAllLines(Paths.get(data));
        } catch (IOException e) {
            e.printStackTrace();
        }
        String csvData = String.join("\n", lines);
        String[] csvLines = csvData.split("\n");
        String[] header = csvLines[0].split(",");
        String[] dataLines = new String[csvLines.length - 1];
```

Code Documentation

/document
/docs

You

/document main.py :

watsonx

```
# Assisted by watsonx Code Assistant
def main():
    """Main function of the script.

Args:
    --data (str): path to input data
    --test_train_ratio (float): test/train split ratio (default: 0.25)
    --n_estimators (int): number of estimators (default: 100)
    --learning_rate (float): learning rate (default: 0.1)
    --registered_model_name (str): model name

Returns:
    None
    """
```

Like Dislike

Q & A

Resources

Copilot lab:

<https://copilot.cloud.microsoft/en-US/prompts>

<https://adoption.microsoft.com/en-us/copilot/success-kit/>

<https://adoption.microsoft.com/en-us/copilot-scenario-library/>

Ivan Portilla

portilla@gmail.com

