F-07.1 - Model Selection (GPT-4 / GPT-3.5) in GitHub Copilot

SUMMARY

In this lesson, we explored how to change the underlying model used by GitHub Copilot and discussed the implications of choosing different models. We began by learning how to access the model selection feature in GitHub Copilot. Depending on your plan—whether it's Pro or Enterprise—the method to enable different models varies.

For GitHub Copilot Pro users:

1. **Access**: Navigate to the `Manage Copilot` settings.
2. **Enable a Model**: You can enable a model by selecting it from the list.
3. **Apply Changes**: Restart VS Code, wait for about five minutes, and you will see the option to choose different models.

For GitHub Copilot Enterprise users:

1. **Access Enterprise Account**: Go to your GitHub Enterprise account settings.
2. **Configure Policies**: Navigate to the policies, find the Copilot section, and select the models you wish to enable according to your organization's policy.

We discussed that the choice of model involves balancing performance, cost, and latency, and we highlighted that selecting the appropriate model can enhance specific use cases. Generally, GPT-4.0 is recommended as a strong default.

WHAT WE LEARNED

- How to enable and switch models in GitHub Copilot.
- The differences between models, focusing on performance, cost, and latency.
- The significance of selecting models based on specific tasks.
- For more comprehensive use cases, different models like Gemini 2.5 Pro can be beneficial.

HOW WE CAN APPLY IT

- **Code Refactoring**: Using GPT-4.5 for complex refactoring tasks.

- **Large Context Tasks**: Employing Gemini 2.5 Pro for extensive document analysis or large codebases.
- **Debugging and Analysis**: Choosing specific models like O3 for debugging complex systems.

TIPS AND TRICKS

- **Default Choice**: GPT-4.0 covers about 98% of common use cases; it's often unnecessary to switch unless your task requires it.
- **Model Latency**: Be aware of latency differences; some models are slower but more suited for specific tasks.
- **Context Window**: Models like Gemini offer large context windows, useful for analyzing extensive documents or codebases.

EXAMPLES

Enabling a Model in GitHub Copilot Pro:

1. Open VS Code
2. Go to Extensions > GitHub Copilot > Manage Copilot
3. Select the desired model (e.g., GPT-4.0)
4. Restart VS Code

Use Case Code Example: Creating a YAML configuration for large datasets:

```
prompt = '''Given this Google Sheets export describing 300 workflows,
generate a YAML configuration file that codifies each workflow
into an orchestration system's format and groups them logically.'''

# Use Gemini 2.5 Pro for better results with large context
result = copilot_api.generate(prompt, model='Gemini 2.5 Pro')
```

Model Suitability Example:

- **Complex Code Generation**: Gemini 2.5 Pro
- **Fast Iterations and Low Latency**: Choose a lower-cost model with quicker response times if your tasks are iterative and frequent.

In essence, model selection in GitHub Copilot can significantly affect the efficiency and effectiveness of your workflows, so understanding each model's strengths is crucial for optimization.