**Desempenho de inteligências artificiais na criação de APIs para plataformas de simulação de investimentos.**

Igor Ayello Borges;Anderson Canale Garcia

**Desempenho de inteligências artificiais na criação de APIs para plataformas de simulação de investimentos.**

**Resumo**

Este trabalho tem como objetivo comparar o desempenho de cinco modelos de inteligência artificial (Copilot, ChatGPT, DeepSeek, Gemini e Meta AI) na criação de APIs para um simulador de investimentos, avaliando métricas como qualidade do código, segurança e eficiência. A pesquisa adotará uma abordagem experimental quantitativa, utilizando ferramentas como Python, FastAPI, Docker e SonarQube. Dados financeiros obtidos de APIs externas serão analisados, e os resultados deverão comparar o desempenho entre os modelos testados.

**Palavras-chave:** Inteligência artificial; Comparação AI; Desempenho SonarQube; Chatbot

**Introdução**

O avanço das tecnologias de extração, armazenamento, transmissão e processamento de dados tem melhorado o desenvolvimento da inteligência artificial nos últimos anos (Carvalho, 2021). Esse progresso possibilitou a criação de novas soluções e produtos, transformando os processos de desenvolvimento. Brandão (2020) destaca que organizações que integram tecnologias de inteligência artificial aos seus processos conseguem atender melhor às demandas de clientes, demonstrando o impacto positivo da adoção dessas ferramentas. Aplicações como o ChatGPT e o GitHub Copilot mostraram capacidade de auxiliar desenvolvedores em tarefas específicas, como a geração de código (Ignácio et al., 2024). Ferramentas de inteligência artificial ainda enfrentam desafios relacionados à precisão e confiabilidade, o que limita sua aplicabilidade em projetos mais complexos.

Entre as ferramentas de inteligência artificial mais populares em 2025 estão o Copilot, ChatGPT, DeepSeek, Gemini e Meta AI, cada uma com características diferentes e modelos específicos. O Copilot e ChatGPT, por exemplo, utilizam modelos da OpenAI, enquanto DeepSeek usa arquiteturas próprias, Gemini e Meta AI são integrados aos ecossistemas do Google e Meta, respectivamente. Essas ferramentas se destacam por sua capacidade de integrar processos e otimizar o trabalho de desenvolvedores.

Nesse contexto, o uso de inteligência artificial no desenvolvimento de software tem se intensificado buscando aumentar a produtividade e a qualidade dos códigos e produtos gerados, contudo, há uma lacuna na avaliação da eficácia de diferentes modelos de inteligência artificial na criação de APIs confiáveis e funcionais. Considerando o crescente uso dessas ferramentas no desenvolvimento de software, é essencial compreender como as tecnologias disponíveis podem contribuir para o desenvolvimento de aplicações específicas, identificando limitações, oportunidades de melhoria, pontos fortes e fracos.

Portanto, este trabalho busca abordar essa lacuna ao comparar o desempenho de cinco modelos de inteligência artificial na geração de APIs para um simulador de investimentos. Ao final, espera-se identificar métricas e tendências que possam contribuir para o aprimoramento do uso de IA no desenvolvimento de software.

**Metodologia ou Material e Métodos**

Será adotada uma abordagem experimental quantitativa para avaliar o desempenho de diferentes modelos de inteligência artificial na criação de APIs destinadas à simulação de investimentos. A pesquisa será desenvolvida utilizando ferramentas tecnológicas específicas e métricas padronizadas para análise comparativa.

Os dados serão coletados de fontes externas confiáveis, como a API YFinance, que fornecerá cotações históricas e dados financeiros do índice Ibovespa, e o site do Banco Central, para obtenção de índices de CDI diário. Esses dados serão organizados e armazenados em tabelas no banco de dados SQL Server, com o objetivo de facilitar a manipulação e a realização de cálculos durante o experimento. Inicialmente, serão selecionados cinco modelos de inteligência artificial amplamente utilizados e reconhecidos em 2025: Copilot, ChatGPT, DeepSeek, Gemini e Meta AI. Cada modelo será utilizado para gerar uma versão de backend de uma API, com funcionalidades voltadas para a importação e análise de dados financeiros, consulta a APIs externas, execução de cálculos de risco e retorno, e armazenamento das estratégias de investimento dos usuários.

Os códigos gerados por cada modelo serão desenvolvidos sob condições uniformes, utilizando a linguagem Python e o framework FastAPI. O Docker será utilizado para criar ambientes de desenvolvimento consistentes, permitindo a virtualização de ferramentas, linguagens de programação e bancos de dados, como o SQL Server. Essa abordagem garante que todas as versões das APIs sejam criadas em configurações semelhantes, permitindo uma comparação justa entre os modelos. A avaliação será realizada utilizando o software SonarQube, que fornecerá métricas detalhadas, como quantidade de bugs, vulnerabilidades, code smells, complexidade do código, desempenho, segurança e qualidade geral. Após a análise inicial das métricas, serão feitos pequenos ajustes nos códigos gerados, visando corrigir erros menores e garantir a validade do experimento. Os dados coletados pelas ferramentas de análise serão comparados quantitativamente, utilizando tabelas e gráficos para identificar padrões de desempenho entre os diferentes modelos de inteligência artificial. A avaliação final buscará destacar os pontos fortes e as limitações de cada ferramenta no desenvolvimento de APIs, oferecendo insights claros sobre sua eficácia e impacto na criação de soluções confiáveis e otimizadas.

**Resultados Preliminares**

As cinco inteligências artificiais cumpriram todos os requisitos e criaram APIs funcionais. Foram realizados os seguintes testes no SonarQube: vulnerabilidades, bugs, code smells (padrões de código), accepted issue (problemas reconhecidos, porém minoritários), duplicações, security hotspots (trechos de códigos que podem representar riscos a segurança), severidade dos bugs, tempo de manutenção, manutenibilidade e complexidade. A seguir apresentarei os resultados detalhados.

Visão geral: Esta seção mostrará a avaliação geral das cinco inteligências artificiais segundo o SonarQube, começarei minha análise com a vulnerabilidade que tem por objetivo verificar as possíveis falhas de segurança nos códigos gerados.

Tabela 1. Vulnerabilidades

|  |  |  |
| --- | --- | --- |
| Inteligência artificial | Número de vulnerabilidades | Avaliação |
| ChatGPT | 1 | E |
| Copilot | 1 | E |
| DeepSeek | 1 | E |
| Gemini | 2 | E |
| MetaAI | 2 | E |

Observando a avaliação de vulnerabilidade de acordo com o SonarQube, o Gemini e MetaAI apresentaram duas vulnerabilidades enquanto ChatGPT, Copilot e DeepSeek uma, todas as inteligências artificiais apresentaram avaliação “E” essa avaliação representa que as cinco avaliadas possuem uma vulnerabilidade bloqueadora, ou seja, uma falha crítica que pode comprometer a segurança do software.

Em seguida analisarei o número de bugs gerados, essa métrica objetiva encontrar possíveis falhas no código que poderão resultar em comportamento inesperado da aplicação, falhas na segurança e dificuldade de manutenção.

Tabela 2. Bugs

|  |  |  |
| --- | --- | --- |
| Inteligência artificial | Número de bugs | Avaliação |
| Copilot | 0 | A |
| DeepSeek | 0 | A |
| Gemini | 0 | A |
| MetaAI | 0 | A |
| ChatGPT | 1 | C |

Com exceção do ChatGPT, as outras inteligências artificiais não geraram bugs e tiveram avaliação A (zero bugs), o único bug gerado pelo ChatGPT teve avaliação “C”, isso significa que o código contém falhas na confiabilidade, mas que não afetam a execução.

Analisarei a métrica “Code Smells” que avalia possíveis problemas de manutenção e qualidade, esses códigos não impedem a execução do software, porém dificultam a manutenção.

Tabela 3. Code Smells

|  |  |  |
| --- | --- | --- |
| Inteligência artificial | Code Smells | Avaliação |
| MetaAI | 5 | A |
| Copilot | 8 | A |
| Gemini | 13 | A |
| ChatGPT | 20 | A |
| DeepSeek | 29 | A |

Todas as inteligências tiveram avaliação “A” indicando que o código está bem estruturado sem problemas significativos de manutenção. Quanto a quantidade de code smells a Meta AI apresentou o menor número (5) e o DeepSeek o maior (29), isso significa que o código gerado pela DeepSeek pode ser melhorado mais vezes em relação a MetaAI gerando mais trabalho futuro para o desenvolvedor como veremos em avaliações futuras.

Quanto as avaliações envolvendo accepted issue (problemas reconhecidos, porém minoritários), duplicações, security hotspots (trechos de códigos que podem representar riscos à segurança) todas as inteligências apresentaram resultados satisfatórios com zero ocorrências e avaliação de segurança “A” considerando que não existem riscos significativos a segurança da informação.

O próximo passo avaliado foi a análise de severidade do código identificando o quão grave é cada severidade sendo: Blocker: Problemas críticos que podem causar falhas graves no software e serão priorizados no momento da correção. Critical: Problemas que representam riscos como, por exemplo, vulnerabilidades de segurança ou erros que comprometem a execução do software. Major: Problemas referentes a qualidade do código e podem diminuir a produtividade dos desenvolvedores. Minor: Problemas que não afetam o funcionamento do código, porém, podem ser melhorados. Info: Sugestões de melhoria. A partir dessas definições tive os seguintes resultados:

Tabela 4. Severidade geral

|  |  |
| --- | --- |
| Inteligência artificial | Severidades encontradas |
| MetaAI | 7 |
| Copilot | 9 |
| Gemini | 15 |
| ChatGPT | 22 |
| DeepSeek | 30 |

Tabela 5. Severidades Bloqueadoras (Blocker)

|  |  |
| --- | --- |
| Inteligência artificial | Severidades bloqueadoras |
| ChatGPT | 1 |
| Copilot | 1 |
| DeepSeek | 1 |
| MetaAI | 2 |
| Gemini | 2 |

Tabela 6. Severidades Críticas (Critical)

|  |  |
| --- | --- |
| Inteligência artificial | Severidades críticas |
| ChatGPT | 0 |
| MetaAI | 0 |
| Gemini | 1 |
| Copilot | 3 |
| DeepSeek | 9 |

Tabela 7. Severidades principais (Majors)

|  |  |
| --- | --- |
| Inteligência artificial | Severidades principais |
| Copilot | 3 |
| MetaAI | 4 |
| ChatGPT | 9 |
| Gemini | 9 |
| DeepSeek | 13 |

Tabela 8. Severidades secundárias (Minors)

|  |  |
| --- | --- |
| Inteligência artificial | Severidades secundárias |
| MetaAI | 1 |
| Copilot | 2 |
| Gemini | 3 |
| DeepSeek | 7 |
| ChatGPT | 12 |

Nenhuma das cinco inteligências gerou severidade do tipo info.

Considerando o ranking geral das severidades a MetaAI gerou menos ocorrências a DeepSeek gerou mais severidades (mais de quatro vezes mais), mesmo gerando menos severidades a MetaAI e o Gemini geraram duas ocorrências cada uma em relação as outras inteligências artificiais, sobre as severidades críticas a MetaAI e o ChatGPT não geraram ocorrências, a DeepSeek gerou nove. Sobre as severidades principais o Copilot foi a ferramenta que gerou menos ocorrências, três, e a DeepSeek gerou mais, treze. Em relação as severidades secundárias a MetaAI foi a que gerou menos, um, e o ChatGPT gerou mais, doze.

O próximo passo foi analisar o tempo estimado para corrigir as severidades sendo, tempo total, falhas de segurança, confiabilidade e manutenção.

Tabela 9. Tempo total

|  |  |
| --- | --- |
| Inteligência artificial | Tempo |
| Copilot | 01h14 |
| MetaAI | 02h07 |
| ChatGPT | 02h27 |
| Gemini | 03h45 |
| DeepSeek | 04h30 |

Tabela 10. Tempo para corrigir bugs de segurança

|  |  |
| --- | --- |
| Inteligência artificial | Tempo |
| ChatGPT | 00h30 |
| Copilot | 00h30 |
| DeepSeek | 00h30 |
| Gemini | 01h00 |
| MetaAI | 01h00 |

Tabela 11. Tempo para corrigir bugs de confiabilidade

|  |  |
| --- | --- |
| Inteligência artificial | Tempo |
| Copilot | 00h00 |
| DeepSeek | 00h00 |
| Gemini | 00h00 |
| MetaAI | 00h00 |
| ChatGPT | 00h10 |

Tabela 12. Tempo para corrigir bugs de manutenção

|  |  |
| --- | --- |
| Inteligência artificial | Tempo |
| Copilot | 00h44 |
| MetaAI | 01h07 |
| ChatGPT | 01h47 |
| Gemini | 02h45 |
| DeepSeek | 04h00 |

A inteligência artificial que gerou menos tempo para ajustes no código foi o Copilot com uma hora e quatorze minutos a que gerou mais tempo foi o DeepSeek com quatro horas e trinta minutos.

Analisando a capacidade de manutenção o SonarQube gerou os seguintes dados: índice de dívida (Debt Ratio), métrica usada dimensionar a dívida técnica quanto maior o índice de dívida mais tempo gasto para correção de bugs, linhas de código, funções, classes e arquivos.

Tabela 13. Índice de dívida técnica

|  |  |
| --- | --- |
| Inteligência artificial | Índice de dívida |
| Copilot | 0,2 |
| MetaAI | 0,3 |
| ChatGPT | 0,4 |
| Gemini | 0,5 |
| DeepSeek | 0,6 |

Tabela 14. Linhas de código

|  |  |
| --- | --- |
| Inteligência artificial | Linhas de código |
| MetaAI | 799 |
| Copilot | 826 |
| ChatGPT | 867 |
| Gemini | 1017 |
| DeepSeek | 1272 |

Tabela 15. Funções

|  |  |
| --- | --- |
| Inteligência artificial | Funções |
| Copilot | 94 |
| Gemini | 95 |
| DeepSeek | 96 |
| MetaAI | 102 |
| ChatGPT | 105 |

Tabela 16. Classes

|  |  |
| --- | --- |
| Inteligência artificial | Classes |
| MetaAI | 16 |
| Gemini | 16 |
| Copilot | 17 |
| DeepSeek | 18 |
| ChatGPT | 22 |

Tabela 17. Arquivos

|  |  |
| --- | --- |
| Inteligência artificial | Arquivos |
| Copilot | 28 |
| MetaAI | 28 |
| Gemini | 29 |
| DeepSeek | 29 |
| ChatGPT | 34 |

A MetaAI gerou o menor número de linhas e classes, o Copilot gerou menos funções e arquivos. Apesar do ChatGPT ter sido o terceiro em linhas totais de código gerado foi a inteligência que gerou mais funções, classes e arquivos.

Finalizando a análise temos os indicadores de complexidade sendo a complexidade ciclomática e a complexidade cognitiva. A complexidade ciclomática é métrica usada para medir a complexidade de um programa com base no número de caminhos independentes que podem ser percorridos no código, quanto maior a complexidade ciclomática, mais difícil pode ser testar, manter e depurar o código. A complexidade cognitiva é usada para avaliar a dificuldade de compreensão do código, quanto maior a complexidade cognitiva maior o risco de erros e mais difícil revisar e refatorar o código.

Tabela 18. Complexidade ciclomática

|  |  |
| --- | --- |
| Inteligência artificial | Complexidade |
| MetaAI | 142 |
| Copilot | 158 |
| ChatGPT | 169 |
| Gemini | 182 |
| DeepSeek | 182 |

Tabela 19. Complexidade cognitiva

|  |  |
| --- | --- |
| Inteligência artificial | Complexidade |
| MetaAI | 68 |
| ChatGPT | 76 |
| Copilot | 87 |
| DeepSeek | 116 |
| Gemini | 144 |

A MetaAI gerou códigos menos complexos nas duas categorias, DeepSeek e Gemini apresentaram códigos mais complexos nas duas categorias.

**Referências**

Ignácio, A.C.; Oliveira, L.S.; Francez, M.P.M.; Eficiência do Uso da Inteligência Artificial no Desenvolvimento de Software. Março 2024.

Carvalho, A.C.P.L.F.; Inteligência Artificial: riscos, benefícios e uso responsável. Abril 2021.

Brandão, R.; Inteligência Artificial, trabalho e produtividade. Novembro 2020.