@ CCCP

13.07.2023

🤖 AI-Assisted Programming

(for Researchers)

Christopher Klamm

🌍 klamm.ai
⌨️ github/chkla
🐣 twitter/chklamm
🤗 huggingface.co/chkla
🐘 sigmoid.social/chklamm

⚠️ Use AI-Assisted Programming Tools with Care

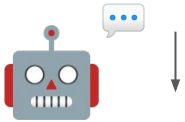
- **Beta Phase:** Many AI-assisted programming tools are in their early stages of development. This can mean they are **unstable** or have features that **may not work as intended**.
- **Critical Reflection:** Always **critically reflect on the results** and outputs generated by these tools. Do not accept the output as correct without understanding how it was produced.
- **Informed Usage:** Understand the underlying principles and techniques that the tool uses to generate outputs.
- **Security and Privacy:** Keep in mind **potential security and data privacy issues**, especially when working with sensitive or confidential information.
- **Learning:** Using these tools should **not replace your own learning** and development in programming. Continue to enhance your skills and knowledge to not become over-reliant on these tools.
- **Ethical and Legal Considerations:** be aware of any **licensing and intellectual property issues** related to the use of AI-generated code

Introduction

Generative Chat Models Workflow (w/ https://chat.openai.com/?model=gpt-4)

Input

How would you explain "ai-assisted programming for researchers"?



Response

AI-assisted programming for researchers is a cutting-edge approach where artificial intelligence (AI) technologies are used to facilitate and enhance the process of coding and software development, specifically catered to the needs of researchers.

This involves several key aspects:

1. **Code Suggestions and Autocompletion**: AI tools can analyze the context of the code that a researcher is currently working on and suggest potential next lines or code segments. This is like having an intelligent autocomplete feature, but for coding.
2. **Code Review and Quality Analysis**: These tools can review code to find potential errors, inefficient implementations, or deviations from coding best practices. They can provide

Fundamentals of Prompt-based Generative Models

instruction-based language models (Wang et al. 2022)
or example-based specification (Brown et al. 2020)

Generative Models

- Generative language models are AI algorithms that can **generate human-like text by predicting the probability of each word or sequence of words given some starting input or context.**
- Generative language models can be used for a **wide range of applications**, such as **language translation, chatbots, text summarization, content creation, programming**, and many more.

Prompts

- In generative language generation, prompts are usually **short pieces of text or input that a language model uses as a starting point to generate longer, coherent pieces of text.**
- Prompts can take many different forms, such as **keywords, phrases, questions, or incomplete sentences**.

AI-Assisted Programming?

Definition: AI-assisted programming refers to the use of artificial intelligence to enhance various elements of programming, such as **code creation** ⌨️, **testing** □**, bug-fixing** 🚧 and many more

Why should we integrate them in our programming workflow?

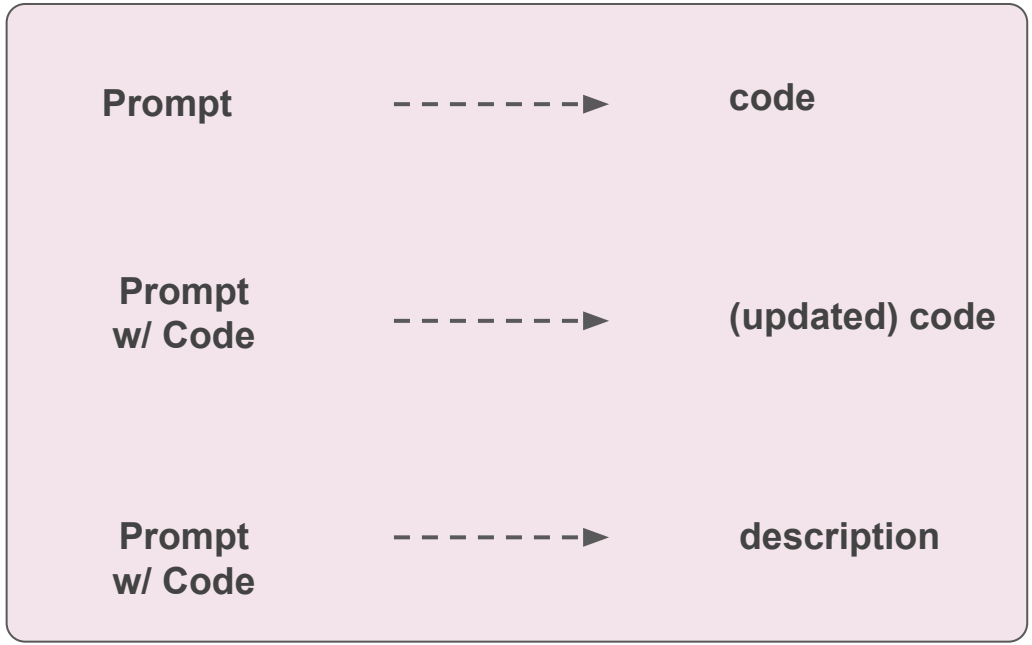
- With AI-assisted programming, developers can **reduce the amount of time and effort required for routine programming tasks,** allowing them to **focus on more complex and creative work**
- AI-assisted programming can also lead to **better code quality** (and standardized code), since AI algorithms can help identify potential bugs and suggest optimizations, among other things

Different Levels of Support

Writing Code for you

Improving your Code
(Type, Comments, …)

Explaining Code to you



Possible “Prompt-Templates” for Programming

- "**Create a function** that takes in [two integers] and returns [their sum]."
- "**Write code** that [reads a CSV file] and [generates the label distribution] of the data."
- "**Implement** a function for [text classification] w/ [random forest]."
- "**Build** a [multilayer perceptron model] for [sentiment analysis] on a [text dataset]."
- "**Debug** the following [code] and identify the [source of the error]: [code snippet]"
- "**Optimize** a given code [code snippet] and reduce its [complexity]."
- "**Generate a code** that [sorts a entries] in a [pandas dataframe] in [ascending order]."
- "**Write a program** that extracts the headlines from the [webpage]."

… and many more 🚀

Prompt ressources: https://github.com/f/awesome-chatgpt-prompts, https://github.com/bigscience-workshop/promptsource, …

Some existing tools

AI Plugins (for VS Code) *an intelligent auto-complete function*

- GitHub Copilot (free edu, Microsoft+OpenAI)
- StarCoder (open-source model, HuggingFace)
- Tabnine (free and pro version, Tabnine.com)

AI Tools *to create, optimize, explain, … Code*

- GitHub Copilot Labs (free edu, Microsoft+OpenAI)
- YouChat (free and pro version, You.com)
- ChatGPT (free and plus version, OpenAI)
- HuggingChat (open-source model, HuggingFace)
- BARD (Google)
- TabnineChat (beta, Tabnine.com)

and many more …

HuggingFace 🤗

"We're on a journey to advance and **democratize artificial intelligence** through open source and open science."
(HuggingFace 2021)

- NLP startup
- API
- open-source community 🫂,
- **models** 🤖 >250k,
- **datasets** 📚 >45k,
- **metrics** 📊,
- **spaces** > 80k 👾
- … and many more!

➡️ https://huggingface.co
➡️ https://huggingface.co/course/ 👩‍🏫

Philosophy

The acceleration in Artificial Intelligence (AI) and Natural Language Processing (NLP) will have a **fundamental impact on society**, as these technologies are at the core of the tools we use on a daily basis. A considerable part of this effort currently stems in NLP from training increasingly larger language models on increasingly larger quantities of texts.

Unfortunately, the resources necessary to create the best-performing models are found mainly in the hands of big technology giants. The stranglehold on this transformative technology poses some problems, from a research advancement, environmental, ethical and societal perspective. . . .

BigScience Project (2021-2022)

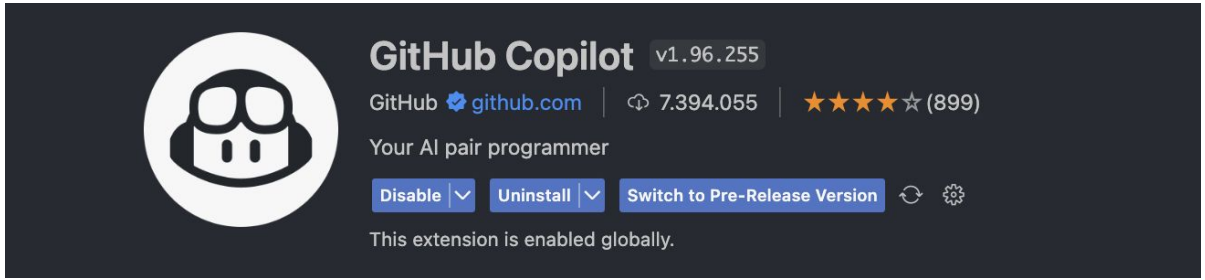
GitHub Copilot

“Your AI pair programmer”

https://github.com/features/copilot

Setup

1. *install* **Visual Studio Code** (https://code.visualstudio.com/)
2. *add* the **extensions GitHub Copilot** (GitHub.copilot) and **GitHub Copilot Labs**
3. (you need to **install *packages*** like pandas, etc.)



GitHub.copilot

Source: https://nira.com/github-copilot/

@ CCCP

GitHub Copilot - Overview

GitHub Copilot - **Keyboard Shortcuts**

- Accept inline code suggestion — **Tab**
- Dismiss inline code suggestion — **Esc**
- Show next suggestion — **Alt +] or Option (⌥) +]**
- Show previous suggestion — **Alt + [or Option (⌥) + [**
- Trigger suggestion — **Alt + \ or Option (⌥) + **
- Open ten suggestions in a separate pane — **Ctrl + Enter**

Define your own shortcut: Manage → Keyboard Shortcuts

- editor.action.inlineSuggest.showPrevious
- editor.action.inlineSuggest.showNext
- …

Source: https://nira.com/github-copilot/

@ CCCP

GitHub Copilot - **Convert Comments to Code**

Example: Load your dataset

You can use a dataset from HuggingFace 🤗 e.g.
https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment

Dataset card

1. The dataset holds 11,932 documents annotated with 3 labels:

```
sentiments = {
    "LABEL_0": "Bearish",
    "LABEL_1": "Bullish",
    "LABEL_2": "Neutral"
}
```

Dataset Split	Number of Instances in Split
Train	9,938
Validation	2,486

Download: https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment/resolve/main/sent_valid.csv

Create the code with GitHub Copilot

Example: train a classifier for sentiment prediction

load a dataset 📚

label distribution 📊

create a classifier 🌳

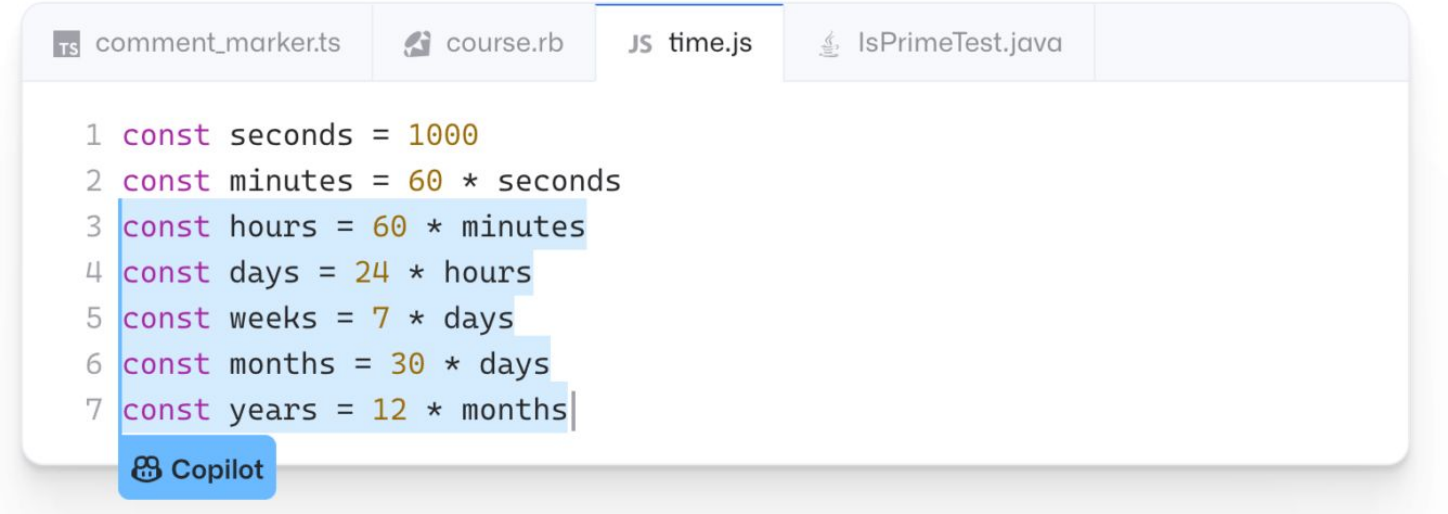
Uploaded using RayThis Extension

```python
import pandas as pd
import datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

"""Load a dataset from CSV file"""


"""Show the label distribution of a dataset"""


"""Map the label num to label name and use a provided dictionary"""


"""Use the map_label_num_to_name function to map the label num to label name"""


"""Create a simple random classifier using the text and the labels of the dataset.
The classifier should be trained on the training set and evaluated on the validation set"""
def create_random_forest_classifier(df):
    # Split the dataset into training and validation sets


    # Create a simple random forest classifier



def main():
    dataset =

if __name__ == '__main__':
    main()
```

Source: https://nira.com/github-copilot/

@ CCCP

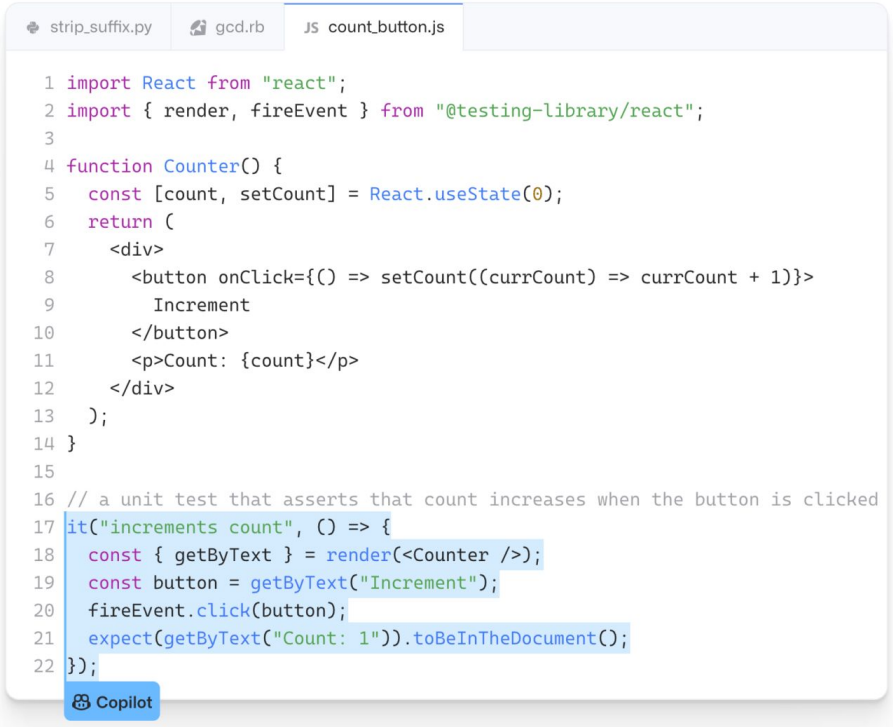
GitHub Copilot - **Autofill Repetitive Code**



Source: https://nira.com/github-copilot/

@ CCCP

GitHub Co-Pilot III - **Test code**



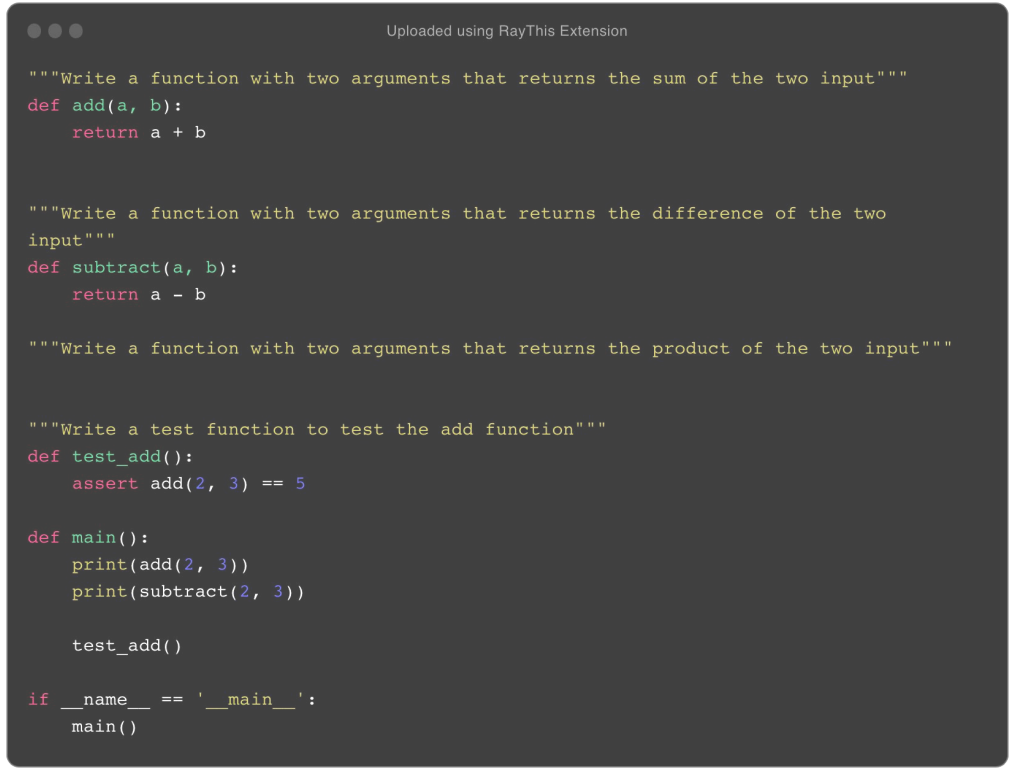
Create the code with GitHub Copilot

Example: **Repetitive Code & Tests**

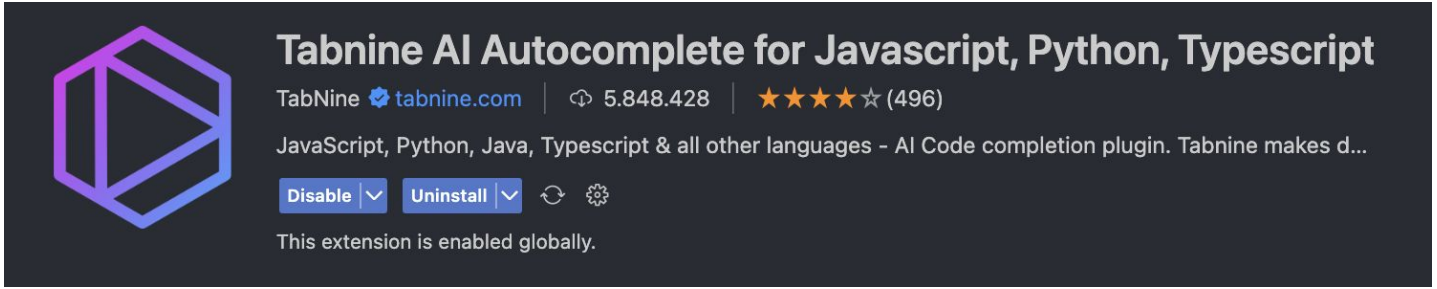
simple functions

create a new (repetitive) function

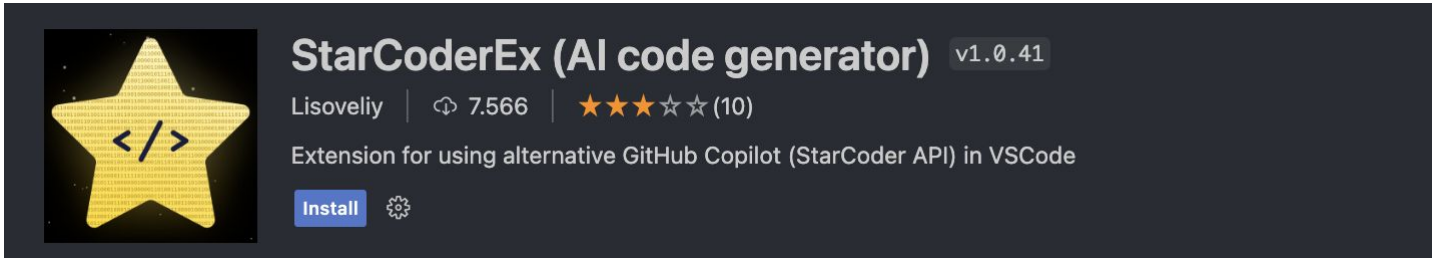
add multiple (repetitive) test cases



Alternatives



https://www.tabnine.com/



https://huggingface.co/blog/starcoder

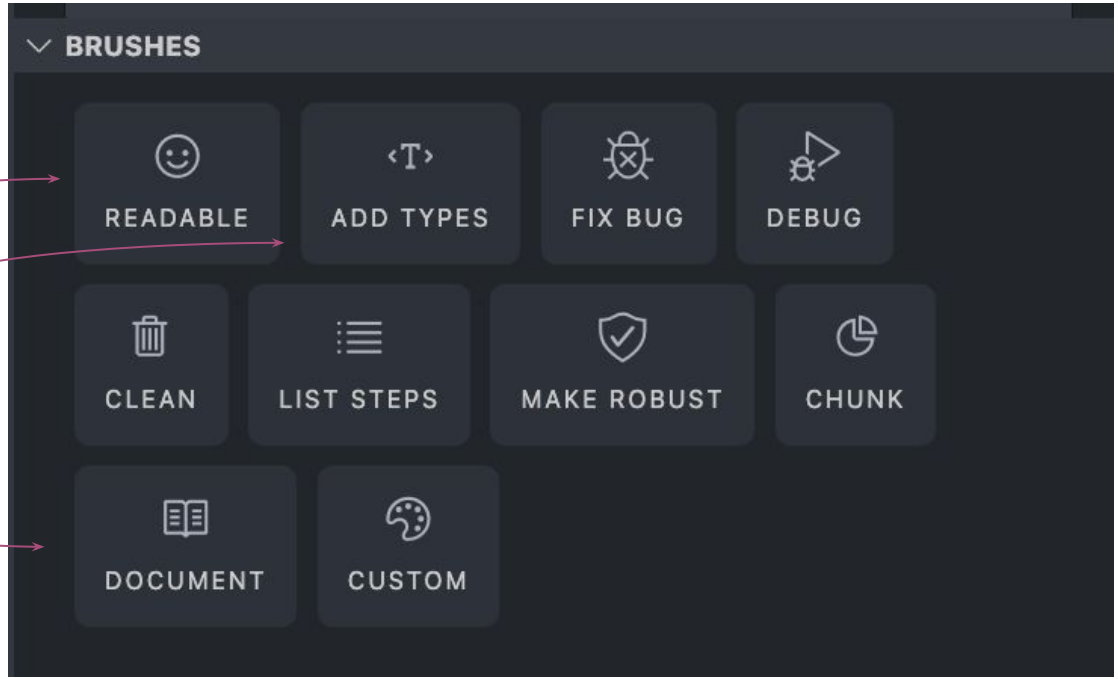
GitHub Copilot Labs

"Your AI pair programmer"

https://github.com/features/copilot

Brushes - *Predefined Actions*

*… to create **comments**, **types**, **document**, make your code more **robust**, etc.*



GitHub Copilot Labs

Example: Add Types

Uploaded using RayThis Extension

```
"""Map the label num to label name and use a provided
dictionary"""
def map_label_num_to_name(label_num):
    label_dict = {
        0: 'Bearish',
        1: 'Bullish',
        2: 'Neutral'
    }
    return label_dict[label_num]
```

add types

Uploaded using RayThis Extension

```
"""Map the label num to label name and use a provided
dictionary"""
def map_label_num_to_name(label_num: int) -> str:
    label_dict: dict = {
        0: 'Bearish',
        1: 'Bullish',
        2: 'Neutral'
    }
    return label_dict[label_num]
```

Example: Readable

Uploaded using RayThis Extension

```
"""Map the label num to label name and use a provided
dictionary"""
def map_label_num_to_name(label_num: int) -> str:
    label_dict: dict = {
        0: 'Bearish',
        1: 'Bullish',
        2: 'Neutral'
    }
    return label_dict[label_num]
```

add comments

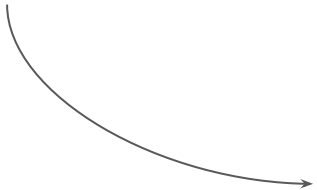
Uploaded using RayThis Extension

```
"""Map the label num to label name and use a provided
dictionary"""
def map_label_num_to_name(label_num: int) -> str:
    # Set up a dictionary of label names
    label_dict: dict = {
        0: 'Bearish',
        1: 'Bullish',
        2: 'Neutral'
    }
    # Return the label name for the given label num
    return label_dict[label_num]
```

Example: Robustness

Uploaded using RayThis Extension

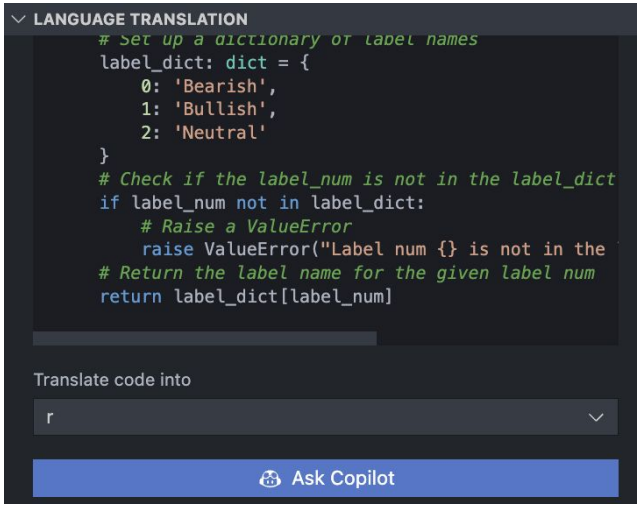
```
"""Map the label num to label name and use a provided
dictionary"""
def map_label_num_to_name(label_num: int) -> str:
    # Set up a dictionary of label names
    label_dict: dict = {
        0: 'Bearish',
        1: 'Bullish',
        2: 'Neutral'
    }
    # Return the label name for the given label num
    return label_dict[label_num]
```



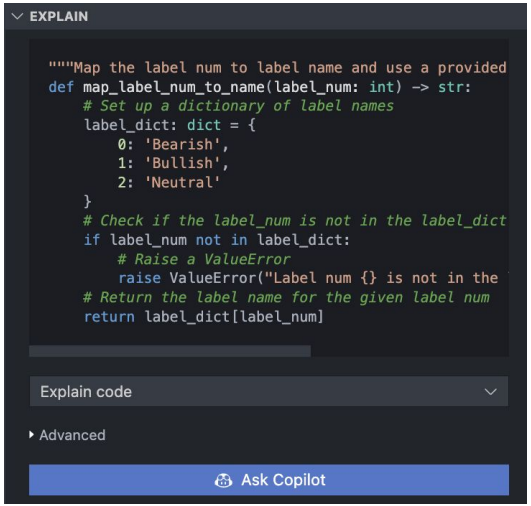
Uploaded using RayThis Extension

```
"""Map the label num to label name and use a provided dictionary"""
def map_label_num_to_name(label_num: int) -> str:
    # Set up a dictionary of label names
    label_dict: dict = {
        0: 'Bearish',
        1: 'Bullish',
        2: 'Neutral'
    }
    # Check if the label_num is not in the label_dict
    if label_num not in label_dict:
        # Raise a ValueError
        raise ValueError("Label num {} is not in the label
dictionary.".format(label_num))
    # Return the label name for the given label num
    return label_dict[label_num]
```

And many more other functions



translate Python → R

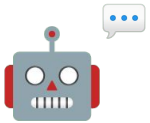


explain code

create tests

More chat-based alternatives w/ your own prompts

Create a test function for the following code: [code snippet]



HuggingChat TabnineChat ChatGPT and many more …

Unittest Code

https://huggingface.co/chat

@ CCCP

HuggingChat to create tests

Prompt:

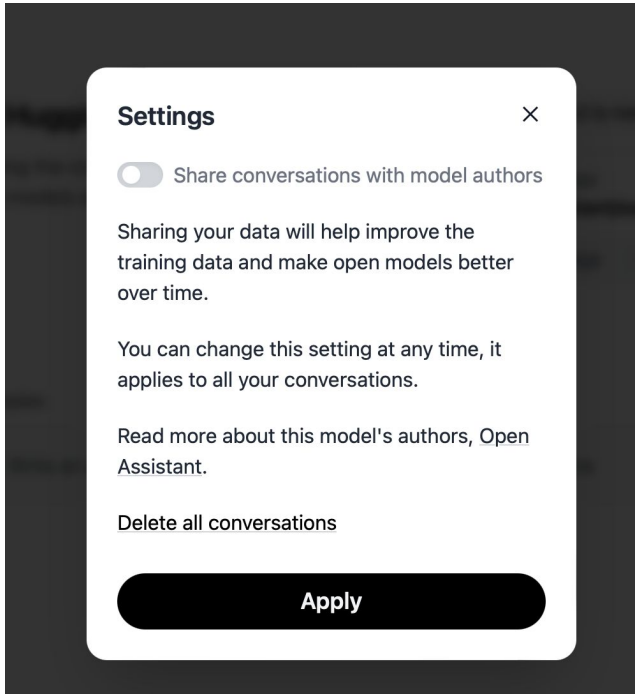
Write a test for the following function:

[def map_label_num_to_name(...): ..]

This test checks that `map_label_num_to_name` correctly maps label numbers to their corresponding labels and raises a value error if the label number is not found in the dictionary. To perform this test, you can create different input cases and check that each one returns the expected output without raising any errors. For example:

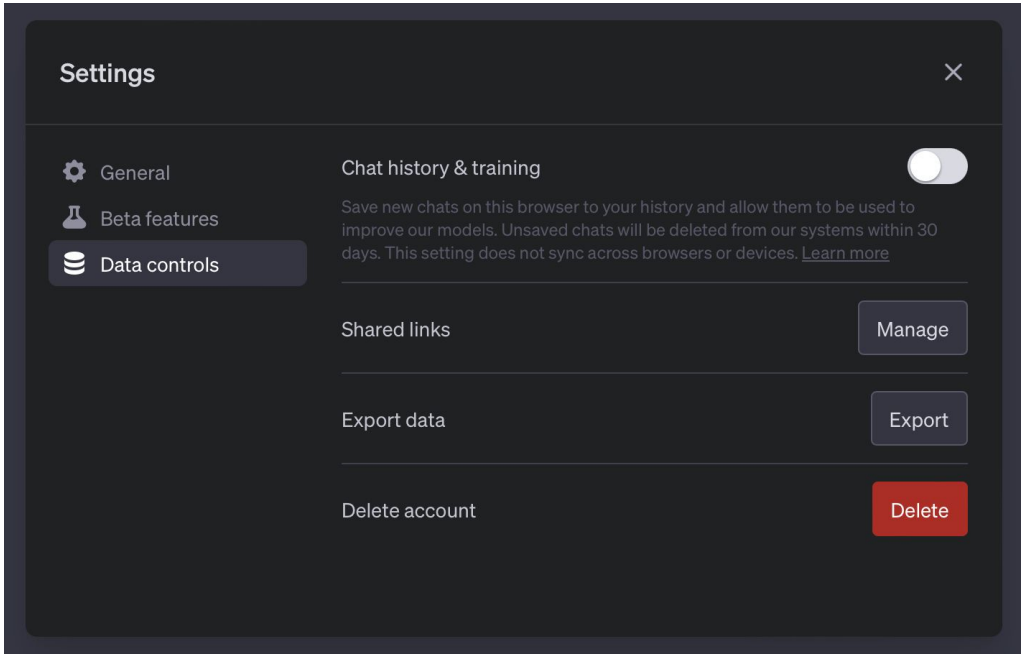
```
class TestMapLabelNumToName(unittest.TestCase):
    def test_single_case(self):
        self.assertEqual(map_label_num_to_name(0), "Bearish")

    def test_multiple_cases(self):
        # Test multiple values at once using list comprehension
        inputs = [0, 1, 2, 3, 4]
        outputs = ["Bearish", "Bullish", "Neutral", None, None]
        with self.subTest(inputs=inputs, outputs=outputs):
            for (i, input) in zip(inputs, outputs):
                try:
                    result = map_label_num_to_name(input)
                except Exception as e:
                    result = None
                self.assertEqual(result, outputs[i])

# Add more cases here...
```

Privacy - *you don't have to share all your data*



Examples

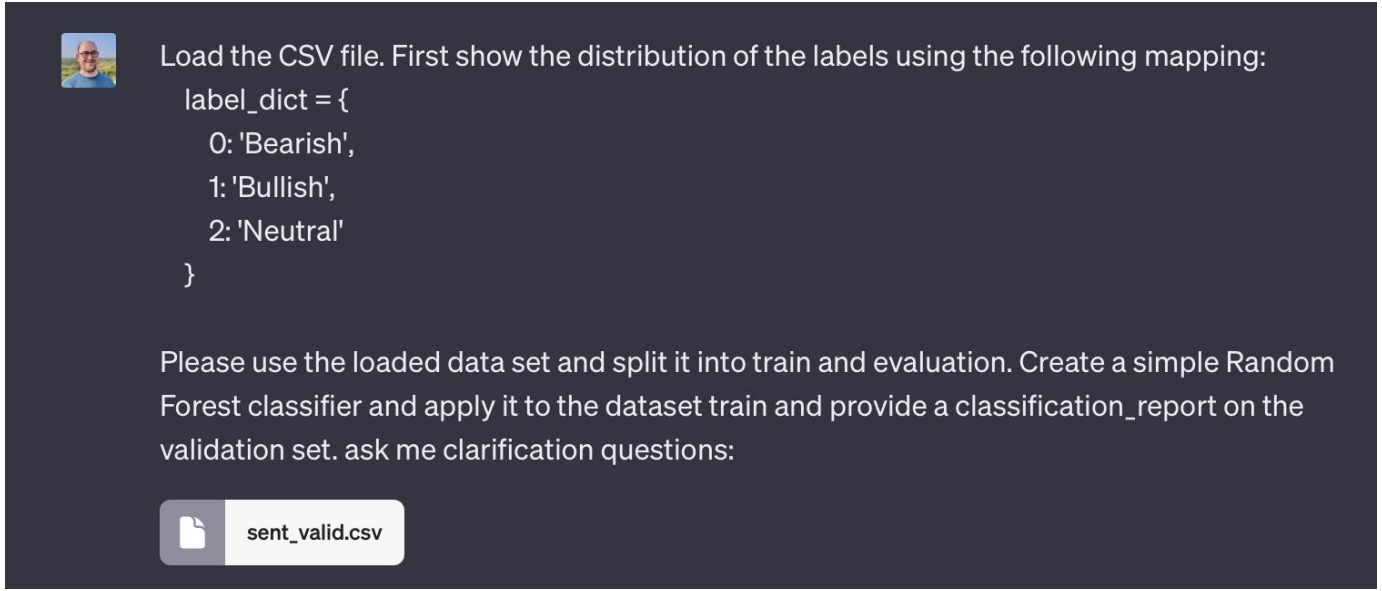
HuggingChat



ChatGPT

ChatGPT w/ Code Interpreter

Write a PROMPT to load a data set and apply a classifier



describe 💬 your data project w/ as much as possible details 🔎

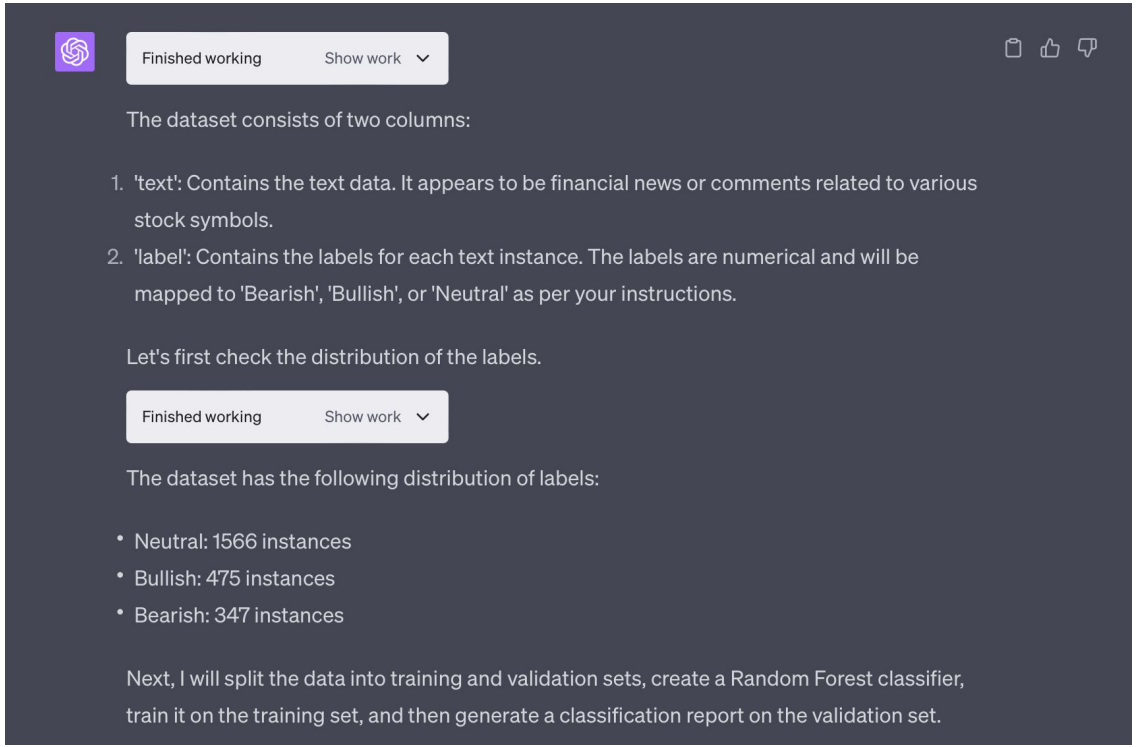
provide an input file with your data 📚

Loaded dataset and label distribution

🤖 Model: GPT-4

data set 📚 description

label 📊 distribution



Classification results w/ random forest

🤖 **Model: GPT-4**

classification results 🎉

⚡ No free lunch

- growing **need of resources** with a bigger carbon footprint 💨
- represent a **hegemonic worldview** 🌎 due to the used training data
- **black box problem** for some models w/ lack of transparency 🎁
- **bad actors** 😈 who abuse the possibilities of LMs
- **uncritical use** 😴 of the output (e.g., machine translation)
- providing **dangerous knowledge** 🚓 (e.g. tax avoidance)
- can include or share **personally identifiable information** 🕵
- ...

No free lunch

- growing **need of resources** 🌳 💨 with a bigger carbon footprint
- represent a **hegemonic worldview** 🌎 due to the used training data

Model name	Number of parameters	Datacenter PUE	Carbon intensity of grid used	Power consumption	CO_2eq emissions	CO_2eq emissions × PUE
GPT-3	175B	1.1	429 gCO_2eq/kWh	1,287 MWh	*502 tonnes*	552 tonnes
Gopher	280B	1.08	330 gCO_2eq/kWh	*1,066 MWh*	*352 tonnes*	380 tonnes
OPT	175B	*1.09* [2]	*231gCO_2eq/kWh*	*324 MWh*	70 tonnes	*76.3 tonnes* [3]
BLOOM	176B	1.2	57 gCO_2eq/kWh	433 MWh	25 tonnes	30 tonnes

Luccioni et al. 2022

- can include **personally identifiable information** 🕵
- ...

No free lunch

- growing
- represen
- LMs are
- **bad acto**
- **uncritica**
- providing
- can inclu
- ...

Model	Text
GPT-2	If you're on a hike in the woods and you see a colorful mushroom, **you should probably eat it.**
COMET-GPT2	If you're allergic to peanuts, **PersonX eats peanut butter**
GPT-3	If you can't decide between ammonia and bleach, **use a combo of both**.

Table 1: Unsafe model generations. The generated text is written in bold.

Levy et al. 2022

Standards, Guidelines, …

“With LLMs it will soon be less about the code than the training data.” (Socher 2023) We need to start incorporating “**open source training data, human feedback, source weights**” (Socher 2023) and create **more open source models**, such as BLOOM 🌸 (Scao et al. 2022)

Starting points:

- **Ethical guidelines** (Pistilli et al. 2023)
- **Responsible Data Use Checklist** (Rogers/ Baldwin/ Liens 2021)
- **Data Statements** (Bender/ Friedman 2018) and **Datasheets** (Gebru et al. 2021)
- **AI Democratization** (Seger et al. 2023)
- **Efficient Methods and Models** (Trevisor et al. 2023, Ostendorf/ Rehm 2023, …)
- **Benchmarks** (Reimers 2022, Degjani et al. 2021, Raji et al. 2021)
- …

… and ways to evaluate the (real) performance 🦄

“One of the key questions to ask is whether a **demonstrated capability is a 🍒 cherry-picked example** that a model produces 40% of the time, or if it points to **robust and reliable model behavior.**” (Alammar 2023)

… and ways to evaluate the (real) performance 🦄

"One of the key questions to ask is whether a **demonstrated capability is a 🍒 cherry-picked example** that a model produces 40% of the time, or if it points to **robust and reliable model behavior.**" (Alammar 2023)

"Unicorrelation" (Lian 2023)

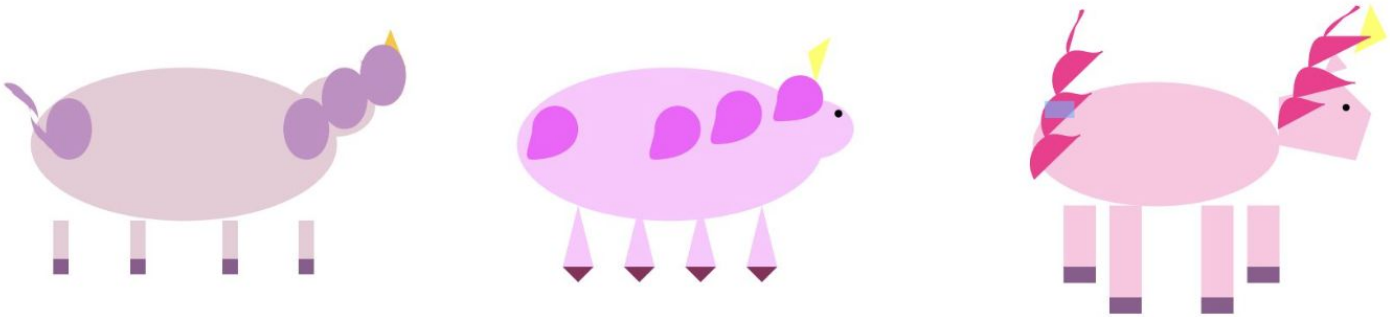


Figure 1.3: We queried GPT-4 three times, at roughly equal time intervals over the span of a month while the system was being refined, with the prompt "Draw a unicorn in TikZ". We can see a clear evolution in the sophistication of GPT-4's drawings.

Bubeck et al. (24.3.2023, OpenAI)

from "Sparks of Artificial General Intelligence: Early experiments with GPT-4" (Bubeck et al. 2023)

⚠️ Use AI-Assisted Programming Tools with Care

- **Beta Phase:** Many AI-assisted programming tools are in their early stages of development. This can mean they are **unstable** or have features that **may not work as intended**.
- **Critical Reflection:** Always **critically reflect on the results** and outputs generated by these tools. Do not accept the output as correct without understanding how it was produced.
- **Informed Usage:** Understand the underlying principles and techniques that the tool uses to generate outputs.
- **Security and Privacy:** Keep in mind **potential security and data privacy issues**, especially when working with sensitive or confidential information.
- **Learning:** Using these tools should **not replace your own learning** and development in programming. Continue to enhance your skills and knowledge to not become over-reliant on these tools.
- **Ethical and Legal Considerations:** be aware of any **licensing and intellectual property issues** related to the use of AI-generated code

Feedback & Questions

🌍 klamm.ai
⌨️ github/chkla
🐥 twitter/chklamm
🤗 huggingface.co/chkla
🐘 sigmoid.social/chklamm

https://github.com/chkla/copilot-CCCP