

# Program Analysis with ML

ML, Coding Companions, The Future

YEGOR BUGAYENKO

Lecture #10 out of 10

90 minutes

All videos are in [this YouTube playlist](#).

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as websites. Copyright belongs to their respected authors.



How Machine Learning Works?

How Program Analysis Fits In?

AI Coding Companions

What's Next?

Chapter #1:

# How Machine Learning Works?

## Deterministic Algorithm

A deterministic algorithm means that given a particular input, the algorithm will always produce the same output:

```
String sayHello(int hours, int minutes) {  
    if (hours > 4 && hours < 12) {  
        return "Good morning!";  
    } else if (hours > 12 && hours < 18) {  
        return "Good afternoon!";  
    } else if (hours > 18 && hours < 22) {  
        return "Good evening!";  
    }  
    return "Good night!";  
}
```

The behavior of the `sayHello()` function is defined upfront by its creator.

## Training a Model

“The process of training an ML model involves providing a learning algorithm with training data. The algorithm finds patterns in the training data that map the input data attributes to the target attributes, and it outputs an ML model that captures these patterns.” (c) Amazon

```
8:35  Good morning!
8:40  Good morning!
10:00  How are you?
11:55  Good afternoon!
13:18  Good day!
14:50  Good afternoon!
15:22  Good afternoon, Sir!
17:14  Good evening!
...
22:34  Evening!
23:50  Good night!
```

$$f : H \times M \rightarrow G$$

$$H = \{0, 1, 2, \dots, 23\}$$

$$M = \{0, 1, 2, \dots, 59\}$$

$$G = \{\text{"Good morning!"}, \\ \text{"Good afternoon!"}, \\ \text{"Good evening!"}, \\ \text{"Good night!"}\}$$

# Embedding

“An embedding is a relatively low-dimensional space into which you can translate high-dimensional vectors.” (c) Google

Features:	Embeddings:	Vectors:
8:35 Good morning!	(8, 30, "morning")	(8, 30, 0)
8:40 Good morning!	(8, 45, "morning")	(8, 45, 0)
10:00 How are you?	NIL	
11:55 Good afternoon!	(12, 00, "morning")	(12, 00, 0)
13:18 Good day!	NIL	
14:50 Good afternoon!	(14, 45, "afternoon")	(14, 45, 1)
15:22 Good afternoon, Sir!	(15, 30, "afternoon")	(15, 30, 1)
17:14 Good evening!	(17, 00, "evening")	(17, 00, 2)
...	...	...
22:34 Evening!	(22, 30, "evening")	(22, 30, 2)
23:50 Good night!	(23, 45, "night")	(23, 45, 3)

## Data Science

There are five tasks to complete repeatedly:

1. Collect and clean a dataset
2. Define features + embeddings
3. Choose and tune an algorithm
4. Train a Model
5. Validate the Model

Chapter #2:

## How Program Analysis Fits In?



# Code Vectorization

...

# Target Attributes

...

Chapter #3:

## AI Coding Companions

[ [Complete](#) PR Review Risks Explain Repeat Tests Refactor LLM ]

## Auto Code Completion

This is how Copilot by GitHub is suggesting code completion in our own programming language, which he definitely hasn't seen before:

```
81 # Converts this to hash
82 [] > as-hash
83 reduced. > @
84 list
85   bytes-as-array ^
86   1
87   [a b]
88   plus. > @
89   times.
90   31
91   a
92   as-int.
93   (0.as-bytes.and b).right 54
```

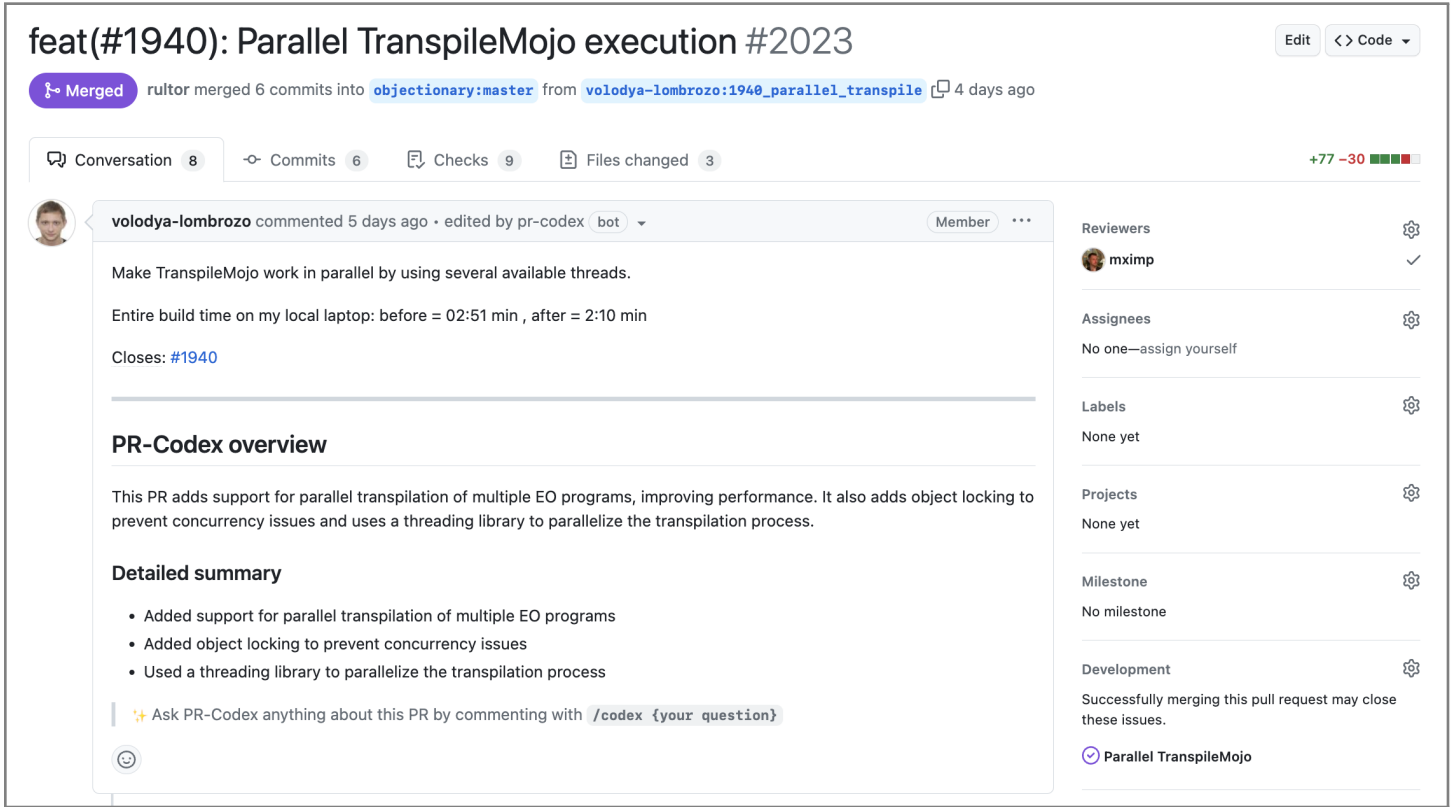
```
81 # Converts this to hash
82 [] > as-hash
83 reduced. > @
84 list
85   bytes-as-array ^
86   1
87   [x y] > +
88     ^,left
89     ^,right
```

```
# Converts this to hash
[] > as-hash
reduced. > @
list
  bytes-as-array ^
  1
  [a b]
```

```
81 # Converts this to hash
82 [] > as-hash
83 reduced. > @
84 list
85   bytes-as-array ^
86   1
87   [a b]
88   plus. > @
89     list
90     a
91     b
```

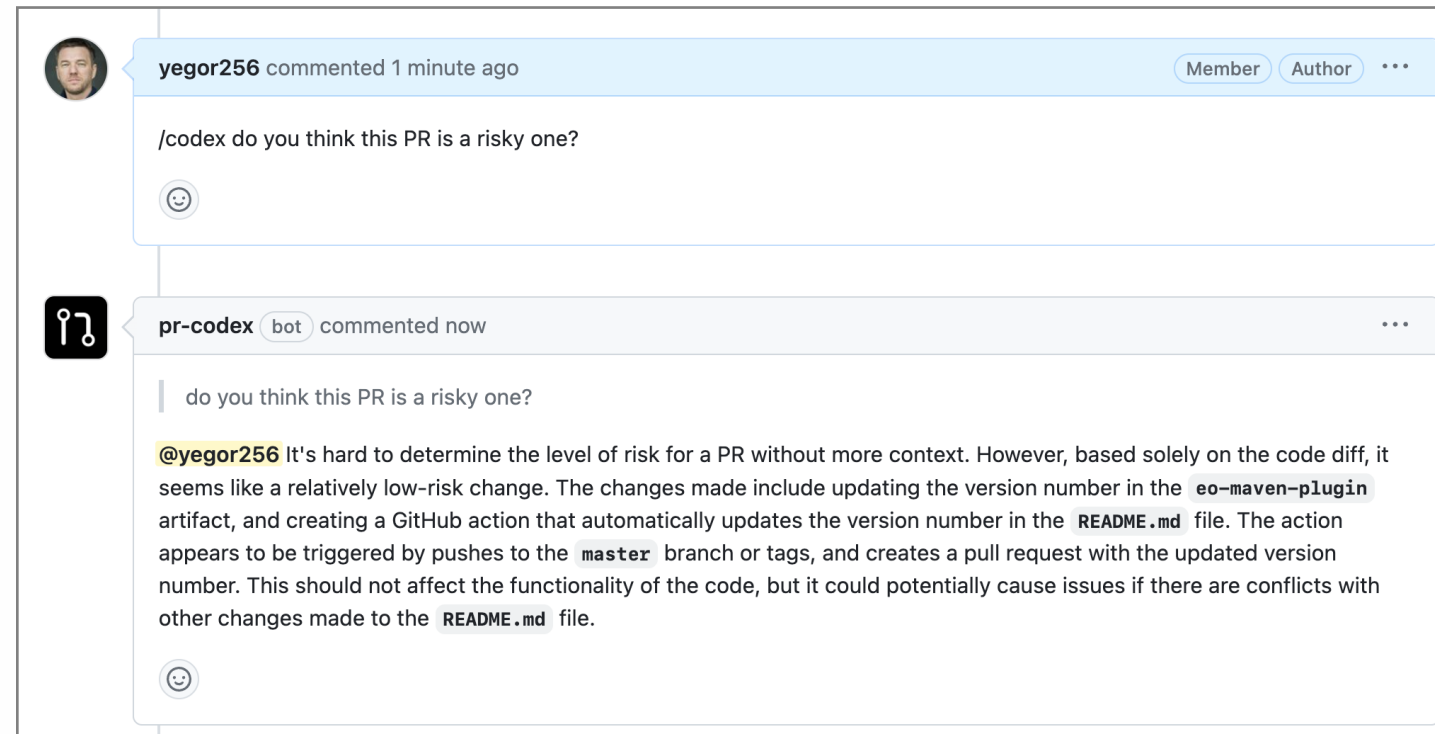
Read also about AWS CodeWhisperer at Werner Vogels' blog. Also, about TabNine (used to be Codota).

# Pull Request Explanation



PR-Codex plugin for GitHub by dlabs

## Reviewing Changes

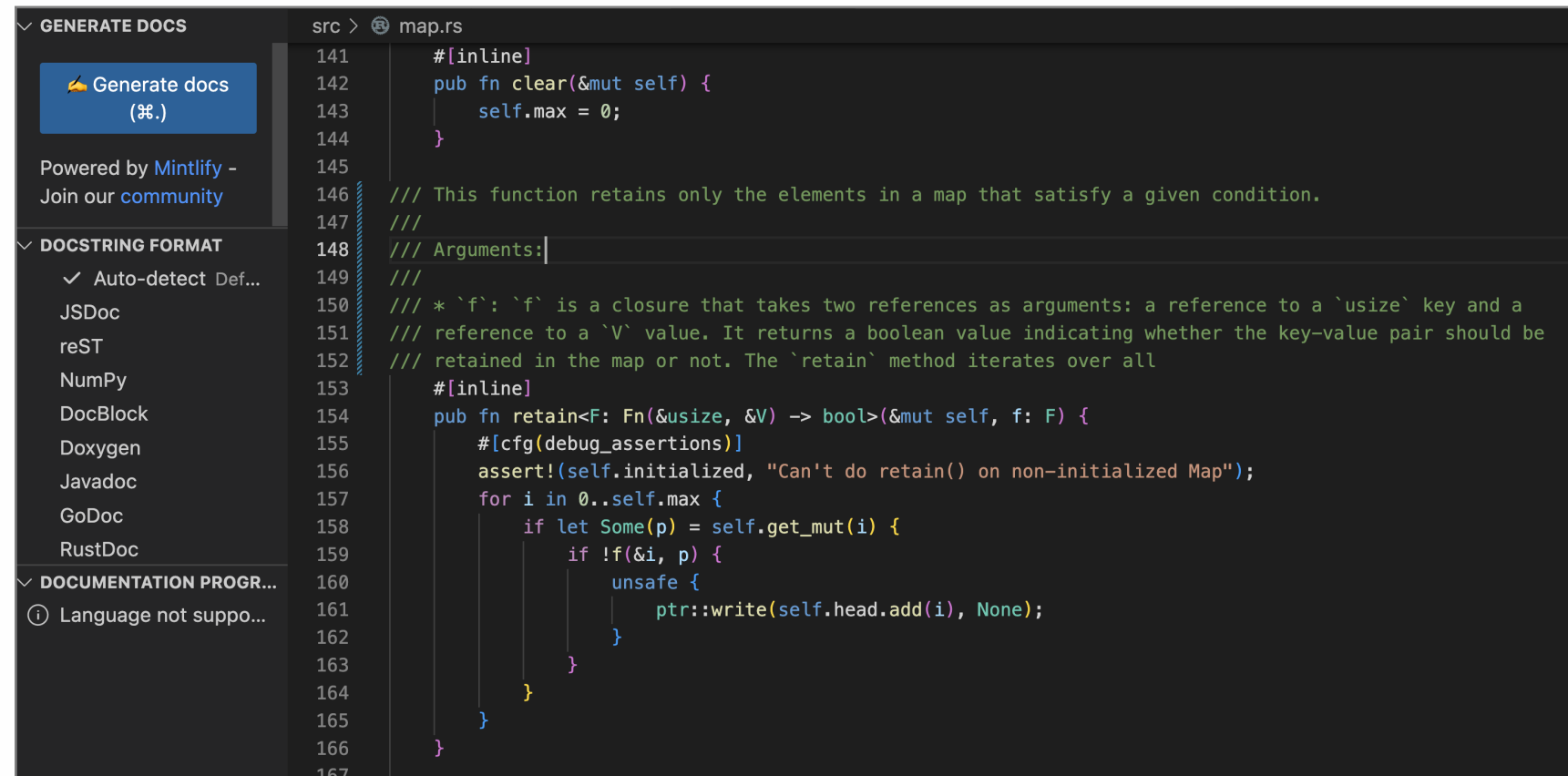


The discussion happened in this GitHub issue: [objectionary/eo:2034](#)

## Pull Request Risk Analysis

[ Complete PR Review Risks Explain Repeat Tests Refactor LLM ]

## Explain This to Me!



```
src > map.rs
141  #[inline]
142  pub fn clear(&mut self) {
143      self.max = 0;
144  }
145
146  /// This function retains only the elements in a map that satisfy a given condition.
147  ///
148  /// Arguments:
149  ///
150  /// * `f`: `f` is a closure that takes two references as arguments: a reference to a `usize` key and a
151  /// reference to a `V` value. It returns a boolean value indicating whether the key-value pair should be
152  /// retained in the map or not. The `retain` method iterates over all
153  #[inline]
154  pub fn retain<F: Fn(&usize, &V) -> bool>(&mut self, f: F) {
155      #[cfg(debug_assertions)]
156      assert!(self.initialized, "Can't do retain() on non-initialized Map");
157      for i in 0..self.max {
158          if let Some(p) = self.get_mut(i) {
159              if !f(&i, p) {
160                  unsafe {
161                      ptr::write(self.head.add(i), None);
162                  }
163              }
164          }
165      }
166  }
167
```

Writer plugin for VS-Code by <https://writer.mintlify.com/>



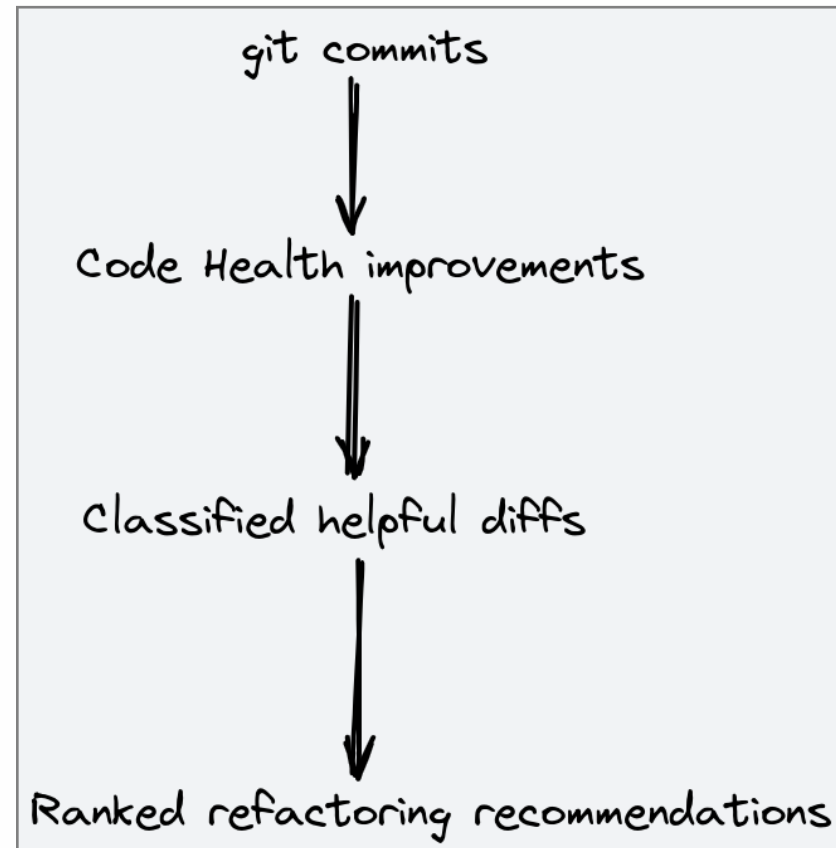
## Repeat After Me!

Read about “making repeated edits easier with IntelliCode suggestions”,  
by Peter Groenewegen.

## Test Case Generation


Ponicode was alive in 2022, now it's ~~dead~~ acquired by CircleCI.

## Automated Refactoring



I found this picture in the [CodeScene website](#).

# Large Language Models (LLM)



Look at this code (it is written in our own object-oriented programming language called EOLANG) and suggest your opinion about the quality of it in terms of readability. Also, if you see any issues, suggest refactoring:

```
...  
[x]>f  
if.>@  
x.lt 2  
x.plus 25  
x.plus 25  
...
```

Based on the code snippet provided, it seems to define a function called `f` that takes an argument `x`, and applies some logic to it based on a conditional statement.

In terms of readability, there are some aspects of the code that could be improved:

1. The use of `>` to indicate the start of the function definition may be confusing to someone who is not familiar with EOLANG. It might be better to use a more descriptive keyword like `function` instead.
2. The use of `if.` and `@` may be confusing to someone not familiar with EOLANG. It's not immediately clear what these symbols mean.
3. The lack of indentation makes it harder to see which statements are part of the conditional block and which are not.
4. The repeated use of `x.plus 25` could be simplified by combining them into a single statement.

Here is a refactored version of the code that addresses these issues:

javascript

Copy code

```
function f(x)  
  if x.lt(2) {  
    x.plus(50)  
  }  
  x
```

In this version, I have used the `function` keyword to define the function, and `if` to define the conditional statement. I have also used curly braces to indicate the start and end of the conditional block, which makes it easier to see which statements are part of the block. Finally, I have combined the two `x.plus(25)` statements into a single `x.plus(50)` statement.

Chapter #4:

**What's Next?**

## Neural Software Analysis

### **“Neural Software Analysis”**

By Michael Pradel, Satish Chandra

Communications of the ACM, January 2022, Vol. 65 No. 1, Pages 86–96

Watch it.

## Electroencephalography (EEG)

### **“Understanding Programming Expertise: An Empirical Study of Phasic Brain Wave Changes”**

By Igor Crk, Timothy Kluthe, Andreas Stefik

ACM Transactions on Computer-Human Interaction, Volume 23, Issue 1,  
Article No. 2, Pages 1–29



Programmers are parts of programs