CLEAN MACHINE LEARNING CODE

Moussa Taifi

Clean Machine Learning Code

Moussa Taifi

This book is for sale at http://leanpub.com/cleanmachinelearningcode

This version was published on 2020-12-26

Contents

Chapter 1 - Clean Machine Learning Code Fundamentals **1**
The Flowchart: Why You Need This Book 1
The Future of Machine Learning Code 1
Bad Machine Learning Code 3
The TCO of a Predictive Service Mess 4
Rebuild the ML Pipelines from Scratch 5
Ideal vs. Real Machine Learning Workflows 6
Taking Responsibility for ML Code Rot 8
Overfitting to Deadlines 9
The Art of Feature Engineering Your Code 10
What Is Clean Machine Learning Code? 10
Inference vs Training of Source Code 12
Active Reinforcement Learning for Source Code 13
Transfer Learning and the Origins of CMLC 13
Conclusion 14
References 15

Chapter 2 - Optimizing Names **16**
Introduction 16
The Objective Function of Names 16
Avoid Mislabeled Labels 16
Avoid Noisy Labels 16
Make Siri Say it 16
Make it Greppable 17
Avoid Name Embeddings 17
Avoid Semantic Name Maps 17
Part-of-Speech Tagging 17
CumSum vs. CummulativeSum 17
Naming Consistency 17
Avoid Paronomasia 17
Use Technical Names 18
Use Domain Names 18
Use Clustering for Context 18
The Scope Length Guidelines 18

Conclusion . . . 19
References . . . 19

Chapter 3 - Optimizing Functions . . . 20
Small is Beautiful . . . 20
3, 4, maybe 5 lines max! . . . 20
Hierarchical functions . . . 20
Single Objective Function . . . 20
Bagging and Function Ensembles . . . 20
Single Abstraction Level . . . 20
Function Arguments . . . 21
Have No Collateral Damage . . . 21
Side-effects in Feature Engineering Pipelines . . . 22
Functional Programming 101 . . . 22
Make Temporal Couplings Explicit . . . 22
Grokking Commands vs. Queries . . . 22
Handling Exceptions . . . 23
Single Entry, Single Exit . . . 23
A Method to the Madness . . . 23
Conclusion . . . 24
References . . . 24

Chapter 4 - Style . . . 25
Comments . . . 25
Don't Hide Bad Code Behind Comments . . . 25
Let Code Explain Itself . . . 25
Useful comments . . . 25
Useless Comments . . . 26
Formatting Goals . . . 27
Python File Size and Notebook Size . . . 27
PEP-8 When You Can . . . 28
Minimize Conceptual Distances . . . 28
One last thing about one-liners . . . 29
Conclusion . . . 29
References . . . 29

Chapter 5 - Clean Machine Learning Classes . . . 30
I Know Classes in Python Why Are You Wasting My Time? . . . 30
Goals for ML Class Design . . . 30
S.O.L.I.D Design Principles for ML Classes . . . 31
Small Cohesive Classes: The Single Responsibility Principle . . . 31
Organizing for Change: The Open-Closed Principle . . . 32
Maintaining Contracts: The Liskov Substitution Principle . . . 32
Isolating from Change I: The Interface Substitution Principle . . . 32

Isolating from Change II: The Dependency Inversion Principle . . . 32
Conclusion . . . 33
References . . . 33

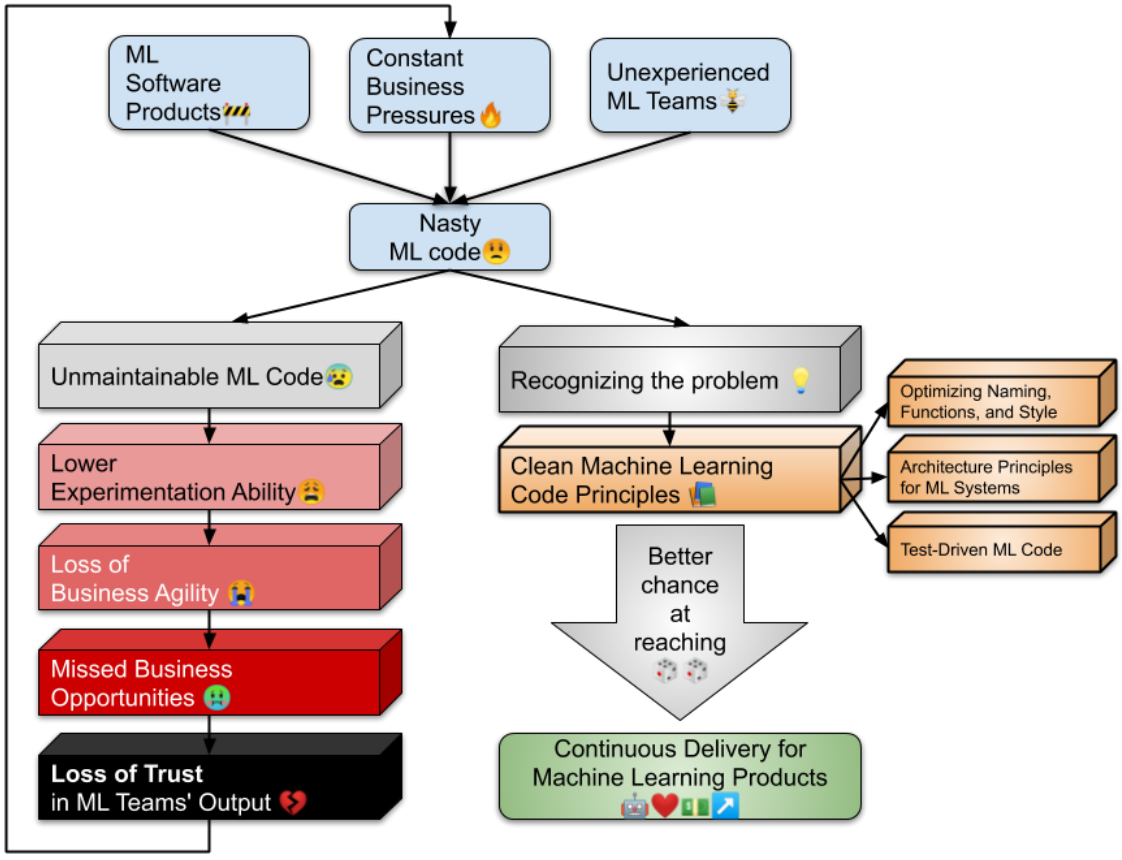
Chapter 6 - ML Software Architecture . . . 34
The purpose of ML Software Architecture . . . 34
Third-party packages are NOT an Architecture . . . 34
Architecture is about Usage . . . 34
Avoiding Chaos using Architecture . . . 34
Frameworks and Harems . . . 34
Defining ML Use-cases . . . 34
Separating High Level Policy from Low Level Implementation . . . 35
The Clean Architecture in One Picture . . . 35
Related Architecture Names and Concepts . . . 35
Friction and Boundary Conditions . . . 36
Taming the Recsys Beast . . . 36
Clean ML Architecture . . . 36
Re-architecting the ML Pipeline . . . 36
Living with a Main . . . 36
Conclusion . . . 37
References . . . 37

Chapter 7 - Test Driven Machine Learning . . . 38
Making Your Life Harder in the Short Term . . . 38
60 Minutes to Save Lives . . . 38
Does ML Code Rot? . . . 38
Tests Let You Clean Your Code . . . 38
Self-testing ML Code . . . 38
What is this TDD you are talking about? . . . 39
Which ML Code Tests Do You Need? . . . 39
GridSearch for ML Code Tests . . . 39
Unit Tests . . . 39
Integration Tests . . . 39
Component Tests . . . 40
End-to-End Tests . . . 40
Threshold Tests . . . 40
Regression Tests . . . 40
Test Implementation techniques . . . 40
Test Doubles . . . 40
Cost Effective Tests . . . 40
Property-based testing . . . 41
Exterminate Non-Determinism in ML Tests . . . 41
The Basics . . . 41

Social Distancing 41
Isolation And Co-mingling 41
The Brave New Async World 41
Working around Remote Services 41
Clocks 42
It Only Fails During Business Hours 42
Test Coverage 42
What To Do If You Are Giving Up on Testing 42
Testing Expeditions a.k.a. Exploratory Testing 42
Synthetic Monitoring 42
Feature Toggles 42
Approaches From Around The ML Community 43
Software 2.0 43
The ML Test Score 43
ML Score Checklist Visualized 43
Coding Habits for Data Scientists 43
Continuous Delivery 43
Conclusions 43
References 44

Chapter 1 - Clean Machine Learning Code Fundamentals

The Flowchart: Why You Need This Book



The Future of Machine Learning Code

We need better machine learning engineers. You are reading this book because you want to be a better machine learning engineer, data scientist, data analyst, or data engineer. Maybe you are also managing teams that are responsible for data analytics or machine learning applications. The whole machine learning industry relies on better machine learning engineers.

In a sense, this book is about rehabilitation from the short-sightedness of quarterly business goals. We must recognize that businesses are addicted to quarterly results. This addiction trickles down to all ranks of software engineers. ML engineers are not immune to this business pressure, and results must come at any price. This book attempts to help the reader identify the role of business pressure on their codebases, and on the mindset and attitudes that lead to ML software disasters. This book also attempts to provide mental and technical tools to kick the habit of mindless code patching to fit the next functional feature into the upcoming launch.

This business pressure is especially relevant for ML engineers that move in the blink of an eye from business explorations to Minimum Viable Product (MVP) to production-level support. For months, these ML engineers kept swinging a bat at a pinata, blindfolded in a dark room; They were trying to improve some obscure offline metric by throwing all the techniques, algorithms, and compute resources at their disposal. It is dark, their arms are tired, and there is candy everywhere on the floor, but we can't see it. Then suddenly they are asked to go to production. This transition can happen without any clear leading signals. Product managers or business leaders ask the engineers to put their MVP in production. The lucky ones know how to move to the expansion phase, which requires them to switch their mindset. Now they are asked for maintainability, reliability, and scalability. They do their best to put their MVP in production. Still, it does not take long before the product managers come up to them infuriated at why a similar functionality that took 3 hours to develop, one month ago, is estimated to take a minimum of 4 weeks to implement in production. The ML developers revise their estimates, and so begins the inexorable downward spiral of ever decreasing code quality and project velocity. This humble book aims at smoothing the transition from exploration to expansion with a set of techniques and principles that proved useful in the software engineering world.

This book will look at ML code from many different dimensions. We will look at the small scale concerns like functions. We will explain useful big ideas of components and architecture design. We will examine the inside details of ML code and the outside form of ML applications. By the end of this book, we will have covered a lot of concepts and coding techniques. You will start recognizing good from bad ML code. You will also learn how to write better ML code and transform smelly, rotten, and disgusting code into fresh and useful code that is easy to change.

Do we need a book about machine learning code? Leading ML tooling providers are bombarding us with managed ML services. They assure us that we are moving away from the old days of coding every single ML application detail. They are saying that we are moving to a new world of AutoML, Automatic Machine Learning Pipeline Generation, and magically managed APIs that will predict your next job, spouse, and favorite book (i.e., like this book). Business folks will be able to speak their minds about what to predict, recommend intuitively, or forecast, and the ML applications will magically appear from thin air.

That is very far from the reality we are experiencing. Here is my prediction. There will be machine learning code, lots of it. Yes, it is reasonable to expect that the level of abstraction will keep rising as the ML industry discovers the right abstractions, libraries, and infrastructure needed. However, humans are not yet able to translate their customer's inspirations and intuitions into working applications without formal, rigorous, and incredibly detailed programs that can be interpreted and

executed by our machines.

Bad Machine Learning Code

Bad machine learning code is among us. But does it matter? Why should we care about this "badness." Why can't we focus on delivering feature after feature? In a way, this whole idea of clean machine learning code relies on a fragile assumption: clean machine learning code matters.

Disasters in ML code are starting to appear all over the place. Bad ML code destroys businesses that had bright futures, and tarnish the image of well-respected companies. Machine learning pipelines are software pipelines, after all. They are full of needless complexity and repetition. They are also very prone to thick opacity, rigidity, and viscosity of design. With these issues, ML failures are growing in importance at an unprecedented pace. Here is a panoply:

- Self-driving cars that are hitting pedestrians.
- Gender bias of large scale translation system.
- Job search engines that are racially biased.
- Simple facial masks that are hacking face id systems in smartphones.
- Machine learning-based social news feeds radicalizing people and contorts public opinions.
- ML generated insurance company premiums predicted to be zero dollars and set for the year.
- Smart financial systems that were making bad decisions and losing $456M in a day (e.g., Knight Capital) because of dead code paths.
- Medical malpractice lawsuits are happening because of unsafe and incorrect treatment ML recommendations.
- Policing systems are prey to ML disasters where facial recognition systems target people of color with high accuracy because that's all they know.

On a less dramatic note, here is an example scenario. A capable junior machine learning engineer with two years of experience started working this month for a medical insurance company. He just inherited an insurance prediction pipeline that predicts optimal premiums for members. Management is asking him to change the model type: ""We want you to change the model from a logistic regression model to a gradient boosted trees model because 'offline' the expanded model works better, and we can increase revenue." No problem, he says, and gives the three weeks minimal estimate. Management is pretty happy with that because they got the minimum estimate.

What kind of issues will he face to get the new version of this critical pipeline working? He clones the project from a git source control system dodging the first bullet. He says to himself, "Nice! Version control will surely help with this project!". What else will he find:

- **Challenge 1**: *Viscosity of design*. He looks for a "tests" folder and finds some notebooks lying around to test the existing code manually.
- **Challenge 2**: *Viscosity of the environment*. He finds that the "notebook unit tests" take around 1 hour to complete.

- **Challenge 3**: *Rigidity*. He ventures to change one name in the loss function tests and finds out that he has to change that variable name in 6 different places.
- **Challenge 4**: *Fragility*. He gets one test related to gradient calculations to work and breaks the code that * generates the test data.
- **Challenge 5**: *Immobility*. He tries to reuse a small data function from the gradient calculations tests to fix the data generation tests, but he is not sure what will be the impact of reusing that function.
- **Challenge 6**: *Opacity*. What does that "ovh_v5" variable name stand for anyway?

Is this a familiar scenario for you? Do you regularly get slowed down and dragged to the border of quitting your job because of a codebase is a jungle of tangled leaky abstractions glued by layers of stitching. We regularly cope and endure, while we attempt to decode encodings, and hope for signs and indications to understand this mad-libs-based story. From the outside, it looks like a well-written story. From the inside, it is a hybrid gallimaufry-hodgepodge that the product and engineering managers want us to extend.

The question is, then, why do You do it this way? Here are the usual options. Maybe you were in a rush to finish that feature engineering pipeline, and you were afraid that your boss would be mad at you for using the time to clean up the code. Maybe you were exhausted by writing the "dumb" model validator and just wanted to move on to a new exciting task. Perhaps you looked at all the promises you made during the previous sprint meeting and felt overwhelmed and started hard-coding models hyper-parameters with index-based variables to get over with this task. Maybe you said to yourself that this hand-crafted model architecture is temporary, and we will fix it later.

Any ML engineer with a couple of projects under their belt did this at one point or another. We all leave messes for later. We tell ourselves that we will come back to this and make it right. We all experienced the comfort of seeing the turtles tower we build work and deciding that it was good enough. We can come back later, add those tests, refactor the functions, and do a deep clean up with some power washing tool to remove the rust, dust, debris, and rubble. On the one hand, you might be thinking, "Don't worry if it doesn't work right. If everything did, you'd be out of a job." - Mosher's Law of Software Engineering. But on the other hand, you vaguely remember the famous saying: "Later equals never."- Le Blanc's law.

The TCO of a Predictive Service Mess

When was the last time messy code impacted your productivity? Were you asked to add a single new categorical feature to an existing ML pipeline only to find that changing anything changes everything?[1]. Over time the mess increases; the entanglement grows. The engineers throw more solutions to the problem. This uncoordinated activity creates more chaos. ML teams that were churning out continuously improving models, now find themselves slower than a tortoise in the sand. Any change requires a full understanding of many other parts of the system. Teams become terrified of changing anything because they don't know the impact of their change. They feel exposed that their code has taken a life of its own. They are frightened by the sheer amount of knowledge

needed to extend anything in the system. The system resists change because of its rigidity. The code is so fragile that changing anything can break non-related functionality. The previous owners left, and the naming conventions are so opaque that no one can clearly explain what a specific function does. The cloak of immobility sets in as the team becomes more helpless.

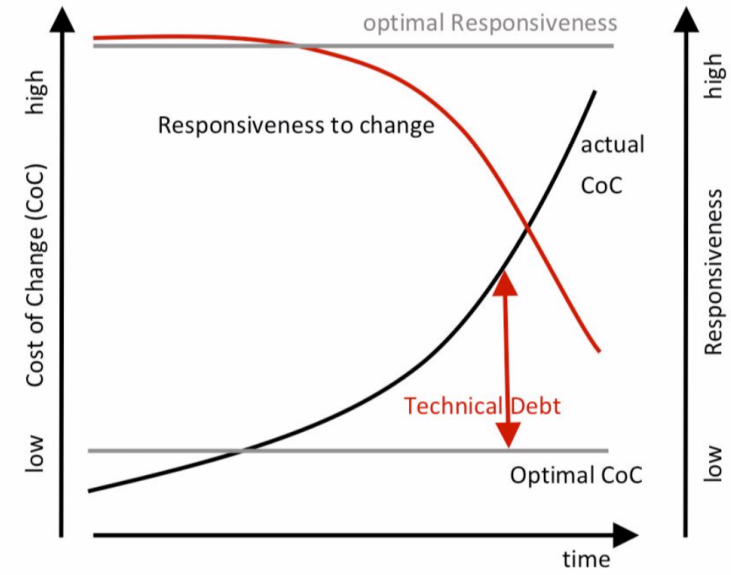


Figure 1. Change in Cost of Change and Responsiveness over time. [6]

Figure 1 shows how the cost of change vs. the responsiveness evolves as time passes. Initially, the productivity/responsiveness of the team is at an all-time high. Management comes up with new functionality, and the ML engineers can explore that path in no time. They come up with a new request: "We need to include the class probability in addition to the existing class prediction to the public API." "No problem, boss, we are on it.". Initially, the cost of a change like this is minimal. "Let me just change the function result to include a second field in the JSON we return to the client…..". As time passes, similar requests start getting delayed, and management notices that the responsiveness is not what it used to be. So they decide the ML team needs more staff to increase productivity. The new team members get onboarded, and suddenly, the project's cost grows in a step function fashion. Now the new team members are getting familiar with the codebase and learn, "How things are done around here." Besides, due to the increase in head-count, the team is under even more pressure to deliver. More mess builds up in the system.

Rebuild the ML Pipelines from Scratch

The ML application cost reaches a tipping point. The V1 team wants to rebuild the ML pipeline from scratch using the latest technology and methodologies. Unfortunately, management knows how much was spent on the first version and is reluctant to unlock the funds for this new pipeline.

They eventually give in to the demands of the ML engineers. A new cross-functional capability-wide team is brought up. The selection of the team members is a nightmare. Everyone wants to be on the

new V2 team, and management selects the best and brightest to staff this newly formed crack team. The rest of the ML engineers are left maintaining the ruins of the existing pipeline. They usually are jealous of the new V2 team. They imagine that working on a greenfield project is all fun and games. But in reality, the two teams are in a race.

The customers still ask for new features, and the v1 team has to ship bug fixes and new functionality. At the same time, the V2 team is looking for documentation on the design of the V1 pipeline, but they don't find any. The only true reference is the system itself. Not only that, but as they learn about the V1 system, the V1 ML engineers keep changing it under their feet.

The rewrite drags on, the V2 team members change over time and morphs into a V3 team. The V3 team looks at the mess created by the V2 team and wants a new redesign because they don't want to maintain the old V2 "new" design.

If you spend any time around software projects, this story is probably familiar to you. For ML engineering, unfortunately, too few reports of such stories exist because of how young the field and the applications are.

The moral of this story is there are many reasons for accepting the cost of keeping ML code clean. The most central reason is professional survival. Not unlike in gardening, few projects can survive without constant attention. ML practitioners learn the hard way that taking the time to care for the living and growing entities that are ML systems is central. Central to both to their mental sanity and the health of their business and its economic goals.

Ideal vs. Real Machine Learning Workflows

With all the hype around machine learning and data science, customers are confused when they are told about the complexity of ML software. They see the following linear figure that defines a nice marketing message of the CRISP-DM:

Figure 2: Idealized ML workflow [2]

Figure 2 shows one such pleasant abstract process with a few well-defined transitions and clearly defined boundaries between ML/DS tasks. The whole image brings a sense of comfort to the reader about the stability and predictability of any ML project. First, there is the business understanding coupled with the data understanding. Then once we have full knowledge of the business data, we move to the data preparation, which is coupled to the modeling part. We repeat the process to get the right data in the right model. Once ready, the best model gets moved to the evaluation part that that will decide if the model is prepared for the business prediction task. This might need us to refine our business understanding and data understanding. Once we are ready to expose the model, we deploy it in a single, nicely defined box of peaceful bliss. The business is happy, the ML engineers rejoice, and everyone gets promoted!

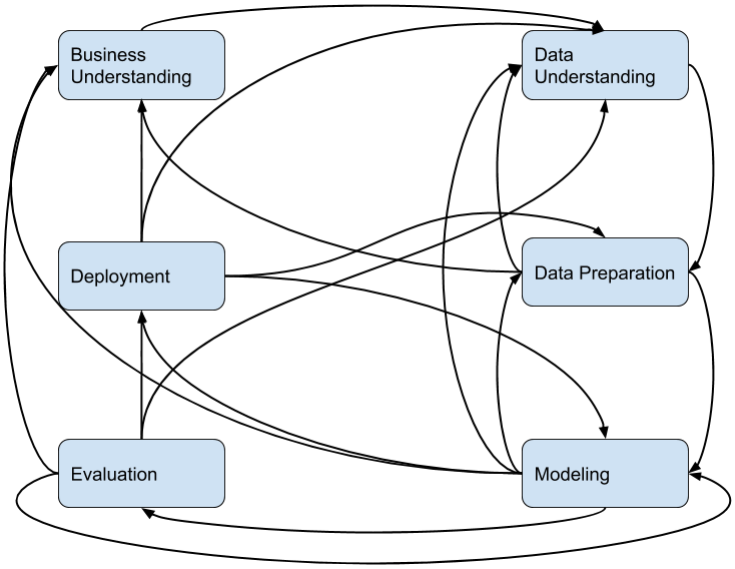


Figure 3: Simplified but Realistic ML tornado workflow

In reality, ML projects, like any software project, are intertwined, domain-specific, engineering-heavy, compromise ridden endeavors. There is no way around that. The marketing and sales process might say otherwise, for good reasons. Still, the reality is that software engineering projects tend to move away from perfect circles and generate tornadoes of dependencies.

This view of ML software can help understand how the code that powers it all will look after a few "iterations." The ML code will reflect the ML software design tornado if left unchecked.

Taking Responsibility for ML Code Rot

Why does working ML code rot? What makes ML code so susceptible to deteriorate and become a nasty mess? We usually look at requirement changes that are not compatible with the current system. We call for the everlasting phrase: "It was not designed to support that functionality." We blame management and leadership for their ignorance of the intricacies of our craft. We get infuriated because of changing schedules and unrealistic timelines. We look at users that change their minds on a whim. All this external blame might be misdirected. The truth is that the fault is most probably coming from us.

I understand that this might get pretty controversial. But we are responsible for guiding the managers, product people, and sales folks. They rely on our judgment to tell them what is possible and what is not realistic. Even when a top-down directive comes crashing from the skies, we should not hold back and be reserved about our opinions on the plans handed to us. The level of responsibility in the hand of an ML engineer is so high that we must speak up. We are just as responsible for ML project failures as any other function on the team.

I hear you say: "Preposterous! I am putting this gross book down!". "If I don't deliver the ML system on time, I'll get kicked out to the curb… I can't stand up to the product manager and tell them that

they are wrong about their estimates … madness!". Most product and engineering managers are reasonable people. They need your honest estimates and opinions. They rely on you to fight back to protect the code with the equal intensity they are fighting to defend their schedules and the promises they made to clients. It is all about checks and balances. They need your unwavering professionalism to help them go fast.

Overfitting to Deadlines

Think about the bias-variance tradeoff. You probably deal with it every now and then when building an ML system. When training ML models, that is a central concept that gets hammered in our heads.

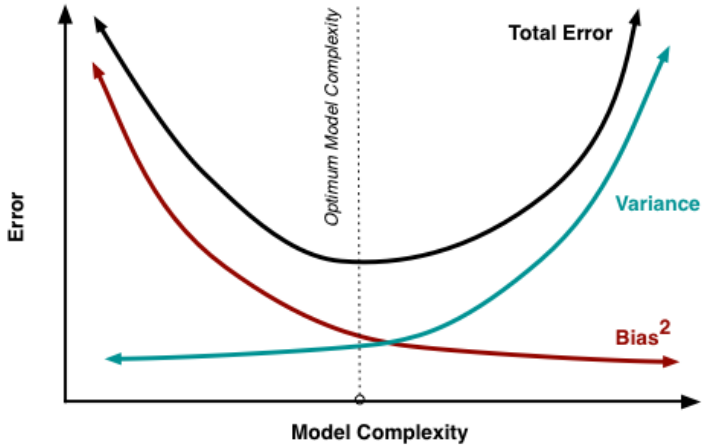


Figure 4 Traditional Bias Variance Tradeoff [3]

High variance leads algorithms to model the random noise of the training data. This usually leads to overfitting. In the software world, this akin to fitting your development style to the deadlines at all costs. Everything takes a back seat compared to the importance of delivering the ML system on time. You get excellent training set accuracy in the short term, but your testing set accuracy tanks dramatically.

High bias, on the other hand, leads algorithms to miss relevant relations between the input features and the predicted output. This usually comes from low complexity models, when we expect, for example, all relationships to be linear. In the software world, this maps to development teams that are stuck with initial assumptions about the software and are not open to change.

Experienced ML practitioners know that systems need to balance their design between underfitting and overfitting to the deadlines. They know that the only way to keep their code manageable is to keep their code clean continuously.

The Art of Feature Engineering Your Code

Feature engineering is still an art form. Coming up with relevant, cheap, and non-correlated features is probably still why ML engineers get paid so well. Experienced ML engineers gather a growing toolbox of feature engineering tricks and techniques that they employ as they see fit. However, when you ask them to explain their reasoning, the conversations usually drifts into, "it worked on a similar problem last year" and "I saw that technique work well in some obscure Kaggle competition write up." They learned the tricks of the trade by trial and error. They built an intuition for what to do when faced with different features.

It is essential to realize that software design is very similar. It is reasonable to ask, "How do I write clean machine learning code?". It is pretty hard to write clean machine learning code when we don't know what it means. Unfortunately, ML engineers write more code than they read. It is similar to asking a screen-writer to improve their character development without having read 10s or 100s of screenplays and novels beforehand.

We are in the same situation; we are asked to write without reading enough. Newcomers to the ML field have a vague sense that some ML code is poorly structured, but they don't have the mental tools to improve the code. They either witnessed full messes or great software. They did not have yet the chance to see live the multitude of transformations, variation, and options that a more experienced ML software writer can perform.

What Is Clean Machine Learning Code?

It should come as no surprise that this book is heavily inspired by the fantastic body of work that Rober Martin provided the software design world. In his book, Clean Code, he recognized that clean code means different things to different people. There are schools of thought, style choices, and philosophies that are very personal to each software professional. How would that translate to machine learning code? Let's see how we can adapt what giants of software design said about software, and attempt to merge it with the goals of machine learning:

- **Robert C. Martin**, the author of the book Clean Code, said, "Remember that code is really the language in which we ultimately express the requirements. We may create languages that are closer to the requirements. We may create tools that help us parse and assemble those requirements into formal structures. But we will never eliminate necessary precision—so there will always be code."
- **Martin Fowler**, the author of the book Refactoring: Improving the Design of Existing Code, famously stated, "Any fool can write code that a computer can understand. Good programmers write code that humans can understand."
- **Michael Feathers**, the author of Working Effectively with Legacy Code, said, "Clean code always looks like it was written by someone who cares."

- **C.A.R. Hoare**, the inventor of quicksort and too many things to mention, said, "The most important property of a program is whether it accomplishes the intention of its user."
- **Donald E. Knuth**, the author of The Art of Computer Programming, said, "The best programs are written so that computing machines can perform them quickly and so that human beings can understand them clearly."
- **Guido van Rossum**, the author of the Python language, said, "Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered."

On the other hand, we have giants of machine learning that express their views on ML design needs:

- **Andrew Ng**, the founder of Coursera and Deeplearning.ai, said, "When you become sufficiently expert in the state of the art, you stop picking ideas at random. You are thoughtful in how to select ideas and how to combine ideas. You are thoughtful about when you should be generating many ideas versus pruning down ideas."
- **Jeremy P. Howard**, Founding Researcher at fast.ai "If you get a detail wrong, much of the time it's not going to give you an exception. It will just silently be slightly less good than it otherwise would have been…You just don't know if your company's model is like half as good as it could be because you made a little mistake."
- **Peter Norvig**, Director of research at Google and author of Artificial Intelligence: A Modern Approach, said: "More data beats clever algorithms, but better data beats more data."
- **Andrej Karpathy**, Director of AI at Tesla, said, "Software 2.0 can be written in a much more abstract, human unfriendly language, such as the weights of a neural network. […] our approach is to specify some goal on the behavior of a desirable program (e.g., "satisfy a dataset of input output pairs of examples", or "win a game of Go"), write a rough skeleton of the code (e.g. a neural net architecture), that identifies a subset of program space to search, and use the computational resources at our disposal to search this space for a program that works."
- **Soumith Chintala**, creator and development lead of PyTorch at Facebook A.I, said "Put researchers first: PyTorch strives to make writing models, data loaders, and optimizers as easy and productive as possible. The complexity inherent to machine learning should be handled internally by the PyTorch library and hidden behind intuitive APIs free of side-effects and unexpected performance cliffs."
- **Martin Zinkevich**, the author of the Rules of Machine Learning and lead on TFX, said, "Plan to launch and iterate: Don't expect that the model you are working on now will be the last one that you will launch, or even that you will ever stop launching models. Thus consider whether the complexity you are adding with this launch will slow down future launches."

One striking element of the above quotes is how intertwined and varied the way knowledgeable people report on their software views. There is a sense of caring for the future of the code by expecting "change" to be a central part of the software life-cycle. There is also a clear sense of excitement in the ML community that can lead to exuberant exploration first mode of operation. There is also a general sense of complexity getting out of hand. These experienced folks lived through

many iterations of software projects and realized that complexity is the enemy. They tell us to use more data instead of sophisticated software. They tell us about the impact of complexity on future launches. They talk about removing side-effects and hiding functionality behind simple APIs. They talk about the difficulties of debugging ML code due to silent errors that stem from complexity. They mention that as the ML developer matures in their craft, they have the mental tools to grow and shrink the code's complexity as needed.

In many ways, this book will be forever imperfect. It is a collection of design techniques that were crowdsourced over the years since the 1970s and beyond. The people that came up with the principles, rules, and laws that we will discuss, have been through the pain of building significant software and were kind enough to express their wisdom so that we don't have to go through the same traps.

One word of caution is necessary here. **None of the elements in this book are absolutely right**. There is no scientific proof that the design techniques in this book are always valid. They are a representation of a school of thought that has good and bad points as all schools of thought have. We hope that it will help you develop a sense of professionalism and craftsmanship by benefiting from our experience. There is ample room for you to disagree with the contents of this book, as you probably will. Some of the materials are radically opposed to the incentives you are trained to seek. We hope you will give this content a look and grab the pieces you need to improve your software as an ML engineer.

Inference vs Training of Source Code

It is estimated that 90% of all the cost of machine learning is at inference time[4]. Generating predictions at inference time far outweighs the cost of producing the trained model. All the optimizations that we perform to improve ML models' training impact at most 10% of the total cost. Since most new projects are geared toward MVP style development, they focus on optimizing the training part and commonly ignore the effort that will be needed in the inference phase. Some less than noble thoughts creep in such as "that will be engineering and operations' problem." and "once I am done with the initial model, I am moving the next project, no need to optimize the inference phase."

I am using this metaphor to relate to reading versus writing software. If you agree with the above inference vs. training cost, you will find that a very similar thing happens on code. We read code a lot more than we write it. You are effectively writing so that other humans can read the code and extend it. We fall into the trap of optimizing the writing of code with no concern for reading it.

Recording screencasts is an excellent way to figure out what happens in that neat Jupyter notebook you are working on. A "writing" session might look something like:

- Open the notebook.
- Scroll down to the last cell where you need to add another aggregation.
- Pause to check if the current dataframe has the correct columns.
- Scroll to the top to where the dataframe was joined with the new index.
- Scroll back down to the last cell.

- Ah! Restore the notebooks cell that was deleted by mistake.
- Get the head of the dataframe.
- Start typing the group by operation.
- Go look up the syntax on the pandas library docs.
- Scroll back up to check that the join was a left join.
- Pause and think.
- Remove the group by that added in the cell.
- Replace it with a simple count that will do the trick this time.

….

Robert Martin, in Clean Code, states it elegantly: “making it easy to read actually makes it easier to write.” You must read the code around the code you are extending. There is not much leeway around that. We must strive to make the code easier to read.

Active Reinforcement Learning for Source Code

Active reinforcement learning is useful when dealing with Sequential Decision Problems. In this setting, the agent is asked to learn and act. It is looking to find an optimal policy to maximize the overall cumulative reward. The core idea is trial and error, but the agent is tuned so that it optimizes the overall “long-term” cumulative reward. Agents are also usually provided with exploration vs. exploitation modes of operation. The exploitation mode helps gather known rewards, and exploration helps the agent discover new ways to increase the total rewards.

These modes are very similar to what we deal with when writing software. We are tasked with writing software to solve a problem faced by the business “right now.” But we are implicitly asked to keep the code base “clean over time.” The same explore exploit strategy can be used for source code. There are times where exploitation and releasing a new functionality is paramount. But there must be time spent exploring and improving the overall code that generates that functionality. Otherwise, the code gets stuck in a local minimum. And we don’t like local minima that much. So the goal is that when you are tasked to change that feature engineering code, pick something small. Fix some naming, break that one hot encoding function into multiple functions, remove a little duplication, and fix that obscure if statement that picks the model to train. Commit back the code just a bit better than you checked it out.

Transfer Learning and the Origins of CMLC

It might change later, but these days, researchers are almost forced to use transfer learning to obtain state-of-the-art results in machine learning. Take NLP tasks; it would be immensely costly to retrain a GPT, BERT, ELMO, or Transformer from scratch. However, if you need to match the performance of published work, then you have no choice. You have to be downstream of the giants of knowledge

compression. You take their model and fine-tune it to fit your use-case. As we get larger and more accurate upstream models, this trend is accelerating.

Similarly, this book is attempting to stand on the shoulders of giants. There is no doubt about it. The awe-inspiring body of work related to software design is breath-taking. Comparatively, software design works that focus on the machine learning field are few and far between. The current 2019 landscape of books and writings focus more on training the current cohort of young, energetic, and immensely ambitious budding ML engineers on the behavior of ML code. The existing training focuses on the behavior of ML code. They focus on how to get started and how to build MVPs. They focus on how to mash together with a set of existing ML tools, and how to duck tape the whole thing with just enough gorilla glue to explore the market for ML products. This is hugely necessary since many of the ideas and inventions will be trashed anyway. This book cares about the next step in the life-cycle of ML software. That moment when there is a hint of a customer that needs your product, and bug fixes and new feature requests start coming in. Recognizing that moment is crucial for the lifetime of the product. In that critical inflection point, the ML engineer should have the mental tools to start training product owners that the effort and time required for exploration and expansion are different. Most importantly, the ML engineer needs to have a plan on how to transition smoothly between the crack-fueled exploration phase, where nothing else matters but product-market fit, and the expansion phase, where stability and maintainability start becoming the core challenge.

This book follows the central philosophy: "To go fast, you must go well" [5]. This fundamental tenet of clean code is the guiding principle of CMLC as well. The challenge is that while CC has guided many cohorts of programmers and still does to this day, the newly minted ML engineers are not aware of what is coming for them. This book aims to act as a reference to train these bright minds on the best practices that the software industry has to offer with examples that machine learning-centric. The concerns we address in this book are geared specifically to the ML crowd, and all the examples target the various dimensions of ML products.

Conclusion

Programmers are akin to studio artists. ML engineers are similar to that as well. They come every day to work and get tasked with creative tasks. Most of them try to balance their artistic needs of organization and elegance, with the business needs of quarterly earnings.

This book is about this art form. But art books usually are only able to share the thought process of fellow artists. This book aims to do the same with a set of programming tricks, principles, techniques, heuristics, disciplines, and methodologies.

Beyond that, it is up to you to take them and include them in your practice. Geoff Colvin put is best in his book, Talent is Overrated: What Really Separates World-Class Performers from Everybody Else, "Deliberate practice requires that one identify certain sharply defined elements of performance that need to be improved, and then work intently on them."

References

[1] Machine learning: The high interest credit card of technical debt. D Sculley, G Holt, D Golovin, E Davydov, T Phillips - 2014 - research.google
[2] https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining
[3] http://scott.fortmann-roe.com/docs/BiasVariance.html
[4] Amazon Elastic Inference: Reduce Learning Inference Cost https://www.youtube.com/watch?v=hqYjkT0BP1o
[5] Clean Code: A Handbook of Agile Software Craftsmanship. Robert C. Martin. 2008. Prentice Hall PTR, Upper Saddle River, NJ, United States
[6] Clean Code Cheat Sheet: https://www.bbv.ch/images/bbv/pdf/downloads/V2_Clean_Code_V3.pdf

Chapter 2 - Optimizing Names

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Objective Function of Names

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Avoid Mislabeled Labels

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Avoid Noisy Labels

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Make Siri Say it

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Make it Greppable

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Avoid Name Embeddings

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Avoid Semantic Name Maps

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Part-of-Speech Tagging

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

CumSum vs. CummulativeSum

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Naming Consistency

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Avoid Paronomasia

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Use Technical Names

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Use Domain Names

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Use Clustering for Context

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Scope Length Guidelines

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Variables

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

References

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Chapter 3 - Optimizing Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Small is Beautiful

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

3, 4, maybe 5 lines max!

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Hierarchical functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Single Objective Function

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Bagging and Function Ensembles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Single Abstraction Level

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

From Input Data to Gradient Calculation and Back

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Function Arguments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Stop at Triadic, Run From Polyadic

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Output arguments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Binary Args

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Clustering Arguments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Positional, Keyword, Args, and Kwargs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Have No Collateral Damage

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Don’t Use Output Arguments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Side-effects in Feature Engineering Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Option 1: Copy everything

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Option 2: Return the new feature column to the caller

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Option 3: Append-only inside the functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Functional Programming 101

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Make Temporal Couplings Explicit

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Grokking Commands vs. Queries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Handling Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Promote The Happy Path With Exceptions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Separate the Happy Path from the Outliers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Don’t Reuse Unrelated Exceptions Types

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Eliminate Duplicates, Doubles, and Homologues

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Single Entry, Single Exit

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Structured Programming 101

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

A Method to the Madness

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

References

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Chapter 4 - Style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Don’t Hide Bad Code Behind Comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Let Code Explain Itself

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Useful comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Legalese Comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Legitimate PSA Comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Why did you do it that way?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Semantic Mapping

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Red Flags Planted in the Sand

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Eventual Consistency with TODO Comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Plugging that Amp

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Docstrings

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Useless Comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Enigmas and Charades

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Split-brain backups

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Fake news

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Mandated Comments / Executive Comments Policy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Journal Comments / Curbing that inner monologue

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Other famous useless comments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Formatting Goals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Use First Impressions To Communicate

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Python File Size and Notebook Size

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

PEP-8 When You Can

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Minimize Conceptual Distances

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Y-Axis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Openness

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Density

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Distance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Ordering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

X-Axis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Horizontal Alignment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Horizontal Openness and Density

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Dedentation and Indentation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

One last thing about one-liners

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

References

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Chapter 5 - Clean Machine Learning Classes

I Know Classes in Python Why Are You Wasting My Time?

Goals for ML Class Design

1. Loose Coupling

2. High Cohesion

3. Change is Local

4. It is Easy to Remove

5. Mind-Sized Components

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

S.O.L.I.D Design Principles for ML Classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Small Cohesive Classes: The Single Responsibility Principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Inversion of dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Class extraction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Facade

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Interface Segregation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Perpetual Engineering Tradeoffs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Organizing for Change: The Open-Closed Principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Great OCP Fiction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Maintaining Contracts: The Liskov Substitution Principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Heuristics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Isolating from Change I: The Interface Substitution Principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Isolating from Change II: The Dependency Inversion Principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

References

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Chapter 6 - ML Software Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The purpose of ML Software Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Third-party packages are NOT an Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Architecture is about Usage

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Avoiding Chaos using Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Frameworks and Harems

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Defining ML Use-cases

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Separating High Level Policy from Low Level Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Clean Architecture in One Picture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Related Architecture Names and Concepts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Concentric Layers Clean Architecture View

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Hexagonal Architecture a.k.a Ports and Adapters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Onion Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Clean Pragmatic Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Friction and Boundary Conditions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Taming the Recsys Beast

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Clean ML Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Re-architecting the ML Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Use Case Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Component View

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Useful Abstractions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Living with a Main

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

References

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Chapter 7 - Test Driven Machine Learning

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Making Your Life Harder in the Short Term

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

60 Minutes to Save Lives

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Does ML Code Rot?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Tests Let You Clean Your Code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Self-testing ML Code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

What is this TDD you are talking about?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Which ML Code Tests Do You Need?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

GridSearch for ML Code Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Unit Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

What Is A Unit?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

To Call Or Not To Call, That Is The Question

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Need For Speed

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Integration Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Component Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

End-to-End Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Threshold Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Regression Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Test Implementation techniques

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Test Doubles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Cost Effective Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Property-based testing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Exterminate Non-Determinism in ML Tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Basics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Social Distancing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Isolation And Co-mingling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The Brave New Async World

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Working around Remote Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Clocks

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

It Only Fails During Business Hours

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Test Coverage

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

What To Do If You Are Giving Up on Testing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Testing Expeditions a.k.a. Exploratory Testing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Synthetic Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Feature Toggles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Approaches From Around The ML Community

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Software 2.0

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

The ML Test Score

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

ML Score Checklist Visualized

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Coding Habits for Data Scientists

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Continuous Delivery

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

Conclusions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.

References

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/cleanmachinelearningcode.