Machine Learning in Production
Midterm 1, Fall 2025

Christian Kaestner and Bogdan Vasilescu

Name: _____

Andrew ID: _____

Instructions:

- Including this cover sheet and the scenario, your exam should have **9** pages. Make sure you are not missing any pages. *You may detach the last two pages and recycle them after the exam.*
- All questions in this midterm refer to the scenario on the last two pages. Answers are graded in the context of the scenario; **generic answers that do not relate to the scenario will not receive full credit.**
- The exam has a maximum score of **57** points. The point value of each problem is indicated. We designed the exam anticipating approximately one minute per point.
- **Please write legibly**. We are unlikely to be able to grade your solution if we can't read it.
- We give an amount of space commensurate with what we expect you to need for each question. We use horizontal lines to suggest where to not use the full page. You may exceed those limits if it is clear where to find the rest of your answer. However, we strongly recommend writing concise, careful answers; short and specific is much better than long, vague, or rambling. However, **do NOT write anything you want us to grade on the back of pages.** We will scan the exam and will not look at the back sides.
- This is a **closed book exam**; no books or electronics allowed. You may refer to 6 sheets of notes (handwritten or typed, both sides). You do not need to hand in those sheets.

Question 1: Goals, Measurement and Telemetry [21 points]

All questions in this exam relate to the DeepCheck scenario on the two last pages. As a first step in the project, you want to make sure that your team is on the same page with what you want to achieve and how to measure whether you are successful.

(a) [3 points] State a *system goal* for the DeepCheck system you are planning to develop. (no measure required)

(b) [3 points] State a *user goal* for the DeepCheck system from the perspective of a state insurance regulator – a government agency that is not buying insurance but is responsible for ensuring fair insurance practices and consumer protection. (no measure required)

(c) [6 points] You want to evaluate a prompt in the agentic system with the specific task of judging whether a weather report found in news search is relevant for a specific claim, similar to how a traditional human insurance fraud investigator would. You want to evaluate this *offline* before you deploy the system. Design a measure, suggest what data to collect, and how to operationalize the measure. The measure can be an approximation, but must be plausible within the realism of the scenario.

Measure:

Data to collect (what and how):

Operationalization:

(d) [6 points] You want to plan how you will evaluate whether the DeepCheck system is helping insurance investigators make accurate fraud decisions *in production*. Design a measure, suggest what data to collect, and how to operationalize the measure *with telemetry*. Be explicit about time frames, if relevant. The measure can be an approximation, but must be plausible within the realism of the scenario.

Measure:

Data to collect (what and how):

Operationalization:

(e) [3 points] After deploying DeepCheck to production, you collect telemetry data on 10,000 potentially fraudulent claims analyzed in the first months. Based on your own telemetry and metric (question above) you want to know whether DeepCheck's results are followed with similar frequency for a slice of small claims (<$5K) and a slice of large claims (>$25K). You run a statistical test comparing the two groups and get p = 0.04.

Which statement is correct? (pick one)

- ☐ There is a 4% probability that DeepCheck's performance is actually the same for small and large claims.
- ☐ There is a 96% probability that DeepCheck performs differently on small versus large claims.
- ☐ If DeepCheck actually performed identically on both claim sizes, we would observe a difference this large or larger in about 4% of random samples.
- ☐ The accuracy difference between small and large claims is practically significant and we should immediately investigate the root cause.
- ☐ We can conclude with 96% confidence that the observed accuracy difference will replicate in future data.

Question 2: Model, Data, and Infrastructure Testing [18 points]

(a) [4 points] To be effective, the LLM in the agentic loop (see scenario) deciding what to search for needs to identify the time and location of the event from the claims. For this you are developing a prompt, based on examples of past claims. You are concerned about overfitting due to *data leakage.* What steps could you take to avoid that problem? (Your answer needs to demonstrate an understanding of the problem of data leakage and must make sense in the scenario).

(b) [4 points] Provide a plausible concrete example of *concept drift* in the scenario (i.e., changes in decision boundaries, not in data distributions) that may degrade the performance of the overall DeepCheck system in production over time.

(writing below this line is allowed but discouraged)

(c) [5 points] One of your model goals is to provide *useful* summaries of the investigation. As a generative task, this is difficult to test offline as there is no single labeled ground truth. Instead, you want to use the idea of *behavioral testing* (aka capability testing, property-based testing). Identify a property to test that relates to usefulness of the summaries and describe how you get test data and labels to determine accuracy for those properties *offline*:

Behavior/property to test:

How to get test data and labels to determine accuracy for this property offline:

(d) [5 points] You expect to frequently modify data preprocessing (e.g., how to extract information from PDFs), external data sources (e.g., APIs for news search), and prompts in the DeepCheck system, so you plan to set up a Continuous Integration system with GitHub actions. You plan to automatically run tests on your data wrangling code and run model evaluations (which you then report to Weights and Biases). You consider using a *self-hosted runner*. What are arguments for and against using a self-hosted runner in this specific scenario?

Main argument for:

Main argument against:

Recommendation for this scenario:
- ☐ GitHub's default runner
- ☐ Self-hosted runner
- ☐ Other:

Question 3: Risks and Mitigation [18 points]

To plan for mistakes you try to better understand the requirements and risks of the product (see the scenario on the last page). For this question, we consider the following *loss* (or harm or risk):

> *"A contractor who does specialized work (e.g., historic homes, energy-efficient "green" buildings) with legitimately higher costs is incorrectly flagged as fraudulent, suffering reputational damage and legal costs from defending themselves."*

(a) [4 points] This loss could have been proactively identified through hazard analysis during system design. Describe the key steps of the hazard analysis process that could have led to identifying this loss and illustrate it for this specific loss:

(b) [3 points] State one **software specification** that, if violated, could cause the loss above.

(c) [3 points] State one **environmental assumption** that designers might make about the real-world context in which DeepCheck operates, but which is *likely not actually true* in practice, and could therefore contribute to the loss above.

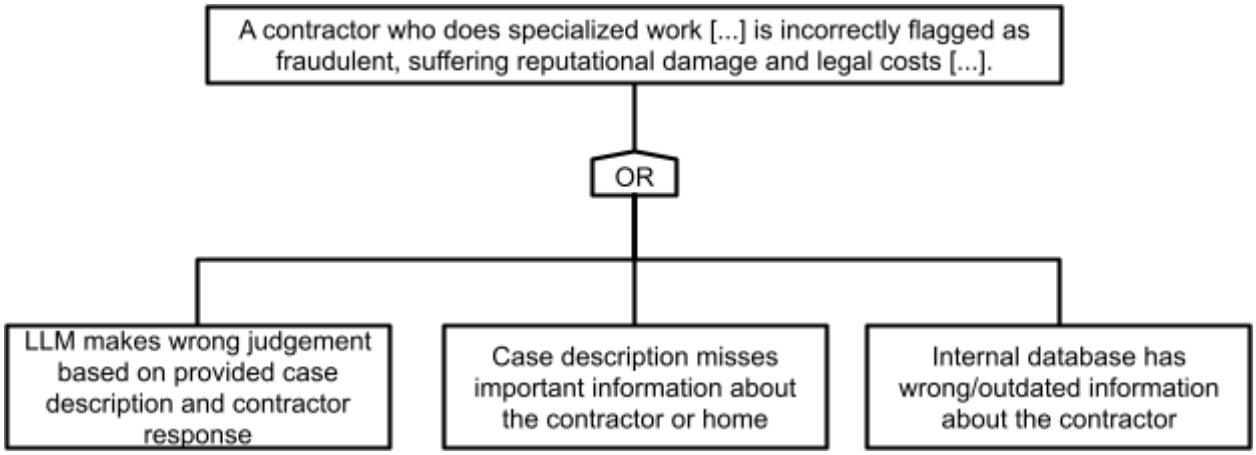
(writing below this line is allowed but discouraged)

(d) [4 points] Despite several attempts at improving the agent prompts, the model still occasionally makes mistakes that could lead to the loss above. You consider the risk as high and want to mitigate the problem to prevent or make it less likely that the loss actually occurs when the agentic system makes a mistake. *The mitigation must be at the system level, outside of improving the ML model or prompts*.

Check one: ☐ The mitigation reduces the likelihood of the loss (improved reliability)
☐ The mitigation prevents the loss (guarantee)

Mitigation description:

(e) [4 points] Update the fault tree below with your mitigation. You can cross out nodes/ connections or add additional nodes/connections as necessary.



Scenario: Agentic Insurance Fraud Detection

You are an ML engineer who just recently joined ClaimGuard AI, an InsurTech company focused on fraud detection based in Boston with 85 employees (most of them working in sales and support). It provides fraud detection as a service to 45 insurance carriers across North America, processing over 2 million insurance claims annually. Of those about 5% are typically fraudulent.

Background: Homeowner insurance protects people from financial losses when their property is damaged. When something goes wrong—a pipe bursts and floods the kitchen, a storm damages the roof, or a fire breaks out—the homeowner files a claim (a formal request for payment) with their insurance company. They submit documents describing what happened, photos of the damage, and a repair estimate from a contractor (a professional who will fix the damage) showing costs. An insurance investigator reviews the claim and decides whether to approve payment. Insurance fraud occurs when someone lies or exaggerates to get more money. Common schemes include: claiming damage that never happened, exaggerating real damage, inflating repair costs, or claiming old damage as new. For example, someone might claim their roof was damaged in last week's storm when no storm occurred, or say repairs will cost $45,000 when typical costs are $20,000. Fraud costs billions annually and raises premiums for everyone.

ClaimGuard's Current System: ClaimGuard started like credit card fraud detection with traditional ML models analyzing numerical patterns (claim amounts, timing, claimant history). Two years ago, they added LLM-powered narrative analysis that reads claimants' written descriptions and identifies red flags like (a) inconsistencies between the claimant's story and contractor's assessment (homeowner describes "minor leak" but contractor lists major structural damage), (b) implausible or overly rehearsed descriptions, and (c) timelines that don't make sense.

These LLM checks run automatically on all claims and successfully flag suspicious patterns. However, they only analyze submitted documents and produce brief summaries. When something suspicious is found, investigators still manually spend hours: searching the internet for weather reports to confirm storms occurred, checking local news for reported events, researching typical repair costs, looking up contractor reputations, and emailing contractors to verify details. This verification work uncovers critical evidence but is slow and expensive.

The New Product: DeepCheck: You have been hired for the *DeepCheck* team, tasked with automating part of that investigation for the claims flagged by the existing system, starting with homeowner insurance claims. Since the investigation is different from claim to claim, human investigators used to have a lot of discretion and flexibility. You want to achieve similar things

automatically and plan to build an *agentic system*, where an LLM decides what actions to take and has access to several tools in a ReAct-style agentic loop. You start with the following tools:

1. *Internet and Database Research:* Searches the web and internal databases to verify claims. Checks news sites and weather databases to confirm events (storms, fires) actually occurred on claimed dates. Looks up typical repair costs in the area to validate contractor estimates. Verifies conditions (were temperatures freezing when pipes allegedly burst?). Searches for patterns in past claims that suggest organized fraud.
2. *Autonomous Emails:* Sends professional verification emails to contractors or repair companies asking targeted questions, waits for responses (48-hour timeout), and incorporates replies into analysis.
3. *Comprehensive Reports:* Produces detailed 2-3 page reports including: original red flags, internet research summaries with source links, email questions and responses, a timeline reconstructing events, evidence supporting or refuting fraud, and recommended next steps.

Example: A homeowner in Austin files a claim on March 20th reporting $45,000 in roof damage from a March 15th hailstorm. They submit a detailed narrative, contractor estimate, and photos. Initial automated analysis flags it as suspicious, triggering DeepCheck, which then searches Austin news/weather for March 15th (finding no hailstorm reported), researches typical Austin roof replacement costs (finding $18,000-$25,000 is normal), searches the contractor's history in an internal database (finds several similar suspicious high-cost claims), emails the contractor: "Can you specify what date you inspected and describe the hail impact patterns you observed?" Finally, after collecting search results and awaiting answers, DeepCheck generates a report: no hailstorm occurred in Austin on March 15th, the contractor never confirmed observing hail damage himself, costs are double typical prices, and this contractor has a suspicious pattern. It recommends denying the claim and investigating the contractor.

Technical constraints: You are working with sensitive data of your customers. ClaimGuard AI uses self-hosted open-weights models as its LLMs. It is built as a cloud-native company with essentially all infrastructure run on AWS cloud servers.

Your Team and Goals: Your team is small, with you and one additional junior ML engineer and two more experienced backend engineers (who help with API, email integration). Agentic technology is new and rapidly evolving, but you are all excited for this project. You can consult with one former fraud investigator now working in sales at ClaimGuard, but getting access to fraud investigators who currently work at your customer insurance carriers has been difficult.

Your team wants to get this feature off the ground as fast as possible as the sales team already starts promising customers huge future time savings in fraud investigations, but there is no fixed release date.

Note: The questions are about the new DeepCheck system for fraud investigation, not the original fraud detection system.