Instructors: Dr. Mikael Lindvall, Dr. Joshua Giltinan
Phone: (301)575 6575
Email: mlindvall@fraunhofer.org

Course Description

Course: ENPM 655– AI-based Software Systems

MARYLAND APPLIED GRADUATE ENGINEERING

Semester: Fall 2022
Day(s): Tuesday
Time: 7:00pm - 9:40pm
Location: JMP 2126

The goal of this new course is to address the important problem of specifying, developing, and testing software systems that are based on artificial intelligence (AI) components. Since such systems are often safety critical or must be dependable for other reasons, it is important that quality is built in throughout the software development life cycle.

It is important to note that the focus of the course is not on generic software engineering or on how to train neural networks, even though we will touch upon those topics. The core of the course is instead about how to specify, develop and test software systems that are based on or uses AI.

Data scientists are often great at building models with cutting edge techniques, but incorporating those models into functioning software products presents different engineering challenges. For example, data scientists may work with un-versioned notebooks on static data sets and focus on prediction accuracy while ignoring scalability, robustness, update latency, or operating cost.

Software engineers, in contrast are typically trained with clear specifications and tend to focus on code, but may not be aware of the difficulties of working with data and unreliable models. They have a large toolset for decision making and quality assurance, but may not know how to apply those to AI-enabled systems and their challenges.

This course discusses questions such as: To what degree can existing SE practices be used for building intelligent systems? To what degree are new practices needed?

This course adopts a software engineering perspective on building intelligent systems, focusing on what a software engineer can do to turn a machine learning idea into a scalable and reliable product.

The course will use software and systems engineering terminology and techniques (e.g., test coverage, architecture views, fault trees) and discuss challenges posed by using such techniques on machine learning/AI components.

The course will include one lecture on teaching/refreshing fundamentals of machine learning and AI to provide a basic understanding of relevant concepts (e.g., feature engineering, linear regression vs fault trees vs neural networks).

The course will also briefly cover design thinking and tradeoff analysis. It will focus primarily on practical approaches that can be used now and will feature hands-on practice with modern tools and infrastructure.

Course Objectives. After completing the course, students will be able to: ·

Analyze tradeoffs for designing production systems with AI-components

· Analyze qualities beyond accuracy such as operation cost, latency, updateability, and explainability

· Implement production-quality systems that are robust to mistakes of AI components

· Design fault-tolerant and scalable data infrastructures for learning models, serving models, versioning, and experimentation

· Reason about how to ensure quality of the entire machine learning pipeline (it should be noted that some of the following topics are still open research questions) with test automation and other quality assurance techniques, including automated checks for data quality, data drift, feedback loops, and model quality.

· Build systems that can be tested in production and build deployment pipelines that allow careful rollouts and canary testing

· Consider privacy, fairness, and security when building complex AI-enabled systems ·

Communicate effectively in teams with both software engineers and data analysts

Assignments include specific deliverables from in-class exercises, participation in the discussion board, and take-home assignments. More information will be provided in the first lecture and throughout the course.

Required technology

Some assignments require access to a computer that runs Python and/or Java, preferably Linux and/or MS Windows 8/10. For some in-class assignments students should bring a laptop to class. More information will be provided during the first lecture.

Method for communication outside the classroom

Email is preferred using canvas.

Important Dates

· See separate schedule

Due dates for Assignments will be provided during the first lecture and throughout the course.

Course attendance policy

Students must attend the following classes: Midterm, Final, Student presentation(s).

Grading procedures

Grading is calculated based on a sum of the weighted scores from (tentative):
Midterm (20%), Final (40%), Assignments (30%), Student Presentation
(10%)

Final information about the grades will be provided during the first lecture. Students who score less than 70 out of 100 on the Midterm must discuss the situation with Dr. Lindvall.

Required/Recommended Textbooks

Textbook is not required. Papers and articles will be handed out as necessary.

More information will be provided during the first lecture.

Course Topics

The following topics are planned to be covered; however, changes may occur:

- Introduction to AI-based Software Systems - Basic Principles and Technologies
  - Basic SE and AI-technologies
  - Tradeoffs among AI Techniques
- Collecting and specifying requirements for systems with an AI component
  - Requirements and Risks
- Designing robust systems with AI components
- Ensuring privacy and other forms of security in AI-enabled systems including legal frameworks for ethics and privacy such as GDPR.
- Software Components and Architecture of AI-enabled Systems
- Testing and Test Coverage Tools for AI-based systems
  - Unit testing data
  - Assuring quality of an AI-enabled system
  - Using test automation to test correctness an AI-enabled system
  - Evaluating correctness or usefulness of a system with an AI component
  - Detecting poor data quality, poor model quality, and data drift
- Reviewing systems with AI components
- How and where to deploy models, how and when to update models, and what telemetry to collect? How to design learning and evaluation infrastructure that scales?