GENERAL INFORMATION:

- Course e-mail: mlcourse@organonanalytics.com
- Office hours: Wednesdays, 14:00-16:00, Organon Office @Etiler
- When: Tuesdays and Thursdays, 18:00-20:30
- Where: Bogazici University, Bilgisayar Muh. binasi A6
- Start Date: 12 November 2019
- Duration: 10 weeks
- Calendar: Slight variations will be possible as per audience demands

There are two overarching goals of this course:

1 Machine Learning Goal(s): Having a good grasp and working knowledge of:

   - ML theory, concepts, terminology
   - Components of an ML pipeline. Technologies, tools, methods, paradigms
   - Building a supervised ML model

2 Numerical Software Development Goal: Developing and maintaining (industrial grade) software is hard. Numerical software adds further complexity as it is memory-intensive, CPU-intensive, and possibly IO-intensive. However, over the decades, the community of computer scientists and software developers have accumulated patterns, principles, and practices that guide the software development process. At the end of the course, the audience will have working knowledge of these (relevant) patterns, principles, and practices (specifically) for numerical software development. In the meantime, he/she will have been involved in the development of a state-of-the-art ML algorithm from scratch.

Accordingly, the course is divided into two parts: In Part-I, We will present the material about Probability, Statistics, Data, Machine Learning, and the GBM algorithm. This material will lay out the theoretical and practical foundation for building Machine Learning models, and for what will be coming in Part-II: The development of a Gradient Boosting Machine algorithm. Many references will be made to the material presented in Part-I.

The syllabus follows:

1 Part-I

1.1 Day-1: Overview, Logistics, Definitions

- Course overview
- Course logistics: Calendar, reference materials, third party resources, etc.
- What is what? Statistical Learning, Machine Learning, Deep learning, Artificial Intelligence: Definitions, commonalities, differences, historical evolution, state-of-the present.

1.2 Day-2: Probability

- Determinism and randomness. Sources of randomness
- Probability Theory as a tool for expressing randomness. Its role in ML.
- Random variables, distributions & densities. Important (discrete and continuous) RVs
- Expectations, conditional expectations, moments of RVs
- Conditional probability, conditional expectation, Bayes' theorem.
- Limit theorems: Law of Large Numbers and Central Limit Theorem
- (Pseudo) random number generation and Monte Carlo simulations

1.3 Day-3&4: Statistics as a prelude to Machine Learning

- Statistics as a tool for understanding data and making inferences from it. Two main branches: Descriptive Statistics and Inferential Statistics. Definitions, examples, use cases
- Univariate (sample) statistics: Mean, variance, quantiles and all that. Multi-modality, outliers, missing values. Examples, code
- Bivariate statistics: Measures of dependency (Linear and non-linear measures). 3 cases: categorical-categorical, categorical-scale, scale-scale. Use cases, examples, code
- The differences between Statistics and Machine Learning? Confusions cleared: a comparison matrix

- The central problem of Statistics and ML: Estimation from finite data. Estimation methods in statistics
- Maximum Likelihood Estimation: The *go-to* estimation method in Machine Learning. A deep dive: formulation, examples
- The measures of the distance (discrepancy) between two distributions (i.e. two random variables). Entropy, cross-entropy. Use cases in ML
- Types of (structured) data in Statistics: Categorical, Ordinal, Scale. Examples
- Hypothesis Testing, Type-I, Type-II errors
- Statistical dependency, correlation and causation. Definitions, pitfalls, examples
- Multivariate Linear Regression: The grandpa of all (supervised) ML algorithms: Specification, estimation, hypothesis testing, feature selection, measuring, goodness-of-fit, prediction
- Ugly facts and how to handle them: Outliers, missing values, categorical variables, non-linearities

1.4 Day-5 Data&Databases

- Data in the wild: structured, semi-structured, non-structured. Definitions, examples. Dataset examples
- Databases: SQL, No-SQL. When to use which.
- SQL basics, and not-so basics. Its role in ML. Major SQL databases.
- Big Data: What is it. The eco-sytem, major tools. When/if you need it.
- Data Quality: Do not take the quality-data granted. Why does it matter? Techniques, examples.

1.5 Day-6: Machine Learning-I

- Machine Learning tasks: Unsupervised, supervised, reinforcement. Definitions, examples, use cases.
- Building an ML model: Training, validation and testing. A recipe will be presented.
- The algorithms you need: GAM, GBM, ANNs. Specifications, examples.

1.6 Day-7: Machine Learning-II

Machine Learning Pipeline:

1 Access and integrate heterogeneous data

2 Measure data quality, take correctiev actions

3 Prepare data (sampling, feature extraction, etc.)

4 Run ML algorithm and build the model

5 Deploy ML model

6 Documenting and monitoring the model

Methods, tools, technologies, examples

1.7 Day-8: GBM algorithm

In the rest of the course, We shall develop a variant of the Gradient Boosting Machines algorithm. So, we need to understand the algorithmic details.

- Derivation of the algorithm. Insight behind it
- Algorithm pseudo-code
- Building a GBM model on an example dataset. Examination of the resulting model
- References: Articles, current software implementations

2 Part-II

In Part-II, we will develop a variant of Gradient Boosting Machines (GBM) algorithm. 12 days will be allocated for this effort. Flashbacks and references will be made to the materials presented in Part-I. The code will be developed with C# using Visual Studio IDE. Set-up instructions will be provided beforehand for time-efficiency. As software development is iterative and incremental, we shall not provide a strict separation of the subject material into distinct sessions. However, a (not-so-strict) ordering of what will be covered is presented below. We will adjust the pace to accommodate all.

- Software development process: Requirements gathering, analysis, design. Iterative and incremental building. Waterfall versus agile. We will discuss and extract an (initial) requirements list for the software. It will be evolved in due time as per the agile method.

- Architectural pattern: We shall adopt Layered Architecture for this software. It is a pattern to manage the complexity at the highest level. We will define what it means in paper, and organize the software modules (as VS projects) to proceed.
- Work at Application Layer: Define shell services that correspond to various steps of building the GBM model. Will serve as the scaffold for the main computation flow.
- Work at Domain(Business) Layer: Discover the primary data structures (and corresponding classes) that will hold the input data needed for the algorithm. Discover the primary and helper classes that will collectively implement the computation as defined by the pseudo-code of GBM.
- Matrix Computations (Infrastructure Layer): Most numerical computing boils down to expressing the CPU-intensive parts of the computation as matrix computations. BLAS and LAPACK libraries are abstractions of these computations. We will teach them. Once you are able to express your data structures and computations in compliant with these libraries, you are (almost) home-free.
- Work at Data Access Layer: How to access data in databases or filesystems and read it into memory. Classes will be developed. Techniques, guidelines, rules-of-thumbs.
- Shared Memory Parallelization (Domain Layer): We will first develop the sequential version of the code. We will then play with various parallelization techniques to parallelize the code so that it works on multi-core chips.
- Work at Application Layer: Connect the Application layer to everything else. Fill out the shell classes created at the outset.
- Validation, Logging, Exception handling: These cross-cutting concerns permeate all of the code. Tools, techniques, and standards will be presented and embedded into the code.
- Testing: System functionality must be tested at the lowest level (unit testing) and at the highest level (integration testing). Will teach you how it is done, and why it is so important.
- Performance: Your should first make the code correct and then optimize it. Numerical software is mostly about optimizing the runtime performance formulated as a function of time, and memory, CPU, I/O consumptions. Will work on performance profiling to understand bottlenecks and how to correct them.