Fin 395 4 Lecture 3:
Machine Learning and Artificial Intelligence

Professor Travis Johnson
The University of Texas at Austin

Objectives for today

1. What is **machine learning**, and how do the core methodologies work?
2. How can we use machine learning in finance & asset pricing?
3. What are some example applications?

4. What is **AI** and how does it relate to machine learning?
5. How can we leverage AI to be better researchers?

Machine learning definition

What is **machine learning**? Gu, Kelly, and Xiu (2020) define it as:

1. "a diverse collection of high-dimensional models for statistical prediction, combined with
2. so-called 'regularization' methods for model selection and mitigation of overfit, and
3. efficient algorithms for searching among a vast number of potential model specifications."

Key difference from standard econometrics: **prediction** ($\hat{y}$) instead of **parametric hypothesis testing** ($\hat{\beta} = 0$)

Machine learning well-suited to some **asset pricing** problems:

1. Lots of data
2. Lots of candidate predictors
3. Measuring moments of data important, less focus on causal effects

Comparison of econometrics and machine learning: terminology

Econometrics	**Machine Learning**
Estimation	Training, learning, fitting
Parameters (β)	Weights, parameters
Regressors, covariates, independent variables	Features
Regressand, dependent variable	Target, label
Model specification search	Hyperparameter tuning
Goodness-of-fit (in-sample)	Performance (out-of-sample)
Computing fitted value	Inference
Statistical inference (e.g., hypothesis tests, CIs)	—
Causal inference (e.g., instruments, DiD)	—
Shrinkage, variable selection	Regularization
Fixed effect	One-hot feature
Objective function	Loss function

Comparison of econometrics and machine learning: philosophy

Concept	**Econometrics**	**Machine Learning**
Objective	Hypothesis testing	Prediction
Feature count		
Parameter count		
Data mining		
Key metrics		
Assumptions		

Overview of core ML methodologies

The ML literature offers a diverse toolkit for empirical researchers

1. **Supervised Learning** (for regression and classification problems)
   - We have X and Y
2. **Unsupervised Learning** (for clustering and density estimation)
   - We only have X
3. **Dimension Reduction Techniques** (often used as pre-processing)
   - We (usually) only have X

Focus today primarily on **supervised learning** methods: ridge regression, LASSO, elastic net, regression trees/forests, boostings, neural networks

Supervised learning problem statement

Goal: Estimate the conditional mean of y_i given a vector of covariates X_i:

$$\mathbb{E}(y_i|X_i) = f(X_i, \theta),$$

where the specification of f and the parameters θ vary across approaches

- This is a canonical problem in both ML and econometrics
- Key differences from traditional econometrics
  - θ can be huge, often larger than the sample size
  - No presumption that the conditional distribution follows a particular parametric model (e.g. $N(X_i\beta, \sigma_i^2)$)
  - Derivatives of conditional expectation w.r.t. any covariate is not of intrinsic interest
  - Focus is on out-of-sample predictions and their accuracy
  - Allow for possibility that conditional expectation may be an extremely non-monotone function with high-order interactions

Penalized linear models

$$\hat{\mathbb{E}}(y_i|X_i) = X_i\hat{\beta}, \qquad \hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - X_i\beta)^2 + \lambda_1 \sum_{k=1}^{K} |\beta_k| + \lambda_2 \sum_{k=1}^{K} \beta_k^2$$

Linear model with penalties applies when β deviates from zero, designed to "shrink" estimates towards an informal prior

- Addresses overfitting, improving OOS performance
- When $\lambda_1 > 0$, often wind up at $\beta_k = 0$ corner solution $\Leftrightarrow$ **variable selection**
  - Helps particularly with 'sparsity' – a bunch of useless variables in X_i
- When $\lambda_2 > 0$, **shrinkage** increasing in deviation from 0
  - Helps particularly with 'colinearity' – some elements of X_i are close to a linear combination of other elements
- **Lasso** (Least Absolute Shrinkage and Selection Operator): $\lambda_1 > 0$, $\lambda_2 = 0$
- **Ridge Regression**: $\lambda_1 = 0$, $\lambda_2 > 0$
- **Elastic Net**: $\lambda_1 > 0$, $\lambda_2 > 0$

Hyperparameters and cross-validation

λ_1 and λ_2 are our first **hyperparameters** – parameters we must specify rather than directly estimate

- Typically passed as parameters in the packages that implement ML algorithms
- Practice of trying different hyperparameters is called **tuning**
- Equivalents in econometrics: number of lags in Newey-West, cluster identifiers, choice of fixed effects, etc.

Cross-validation approach splits sample into three:

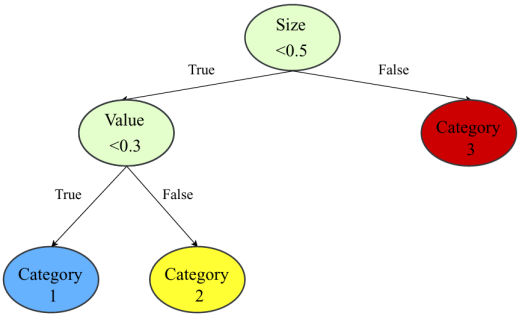
1. **Training**: estimate θ given a set of hyperparameters λ
2. **Validation**: iterate over possible λ values, maximizing performance in this period using forecasts from θ estimated in training period
3. **Test**: measure performance here using the λ, θ estimated using prior periods

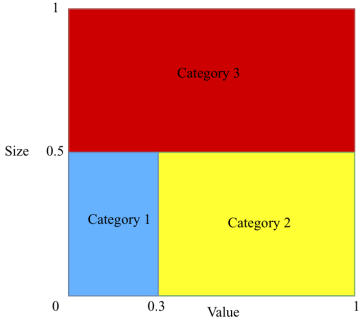
For time-series/panel data in asset pricing, we often re-estimate θ, λ each year and use the subsequent year as the test sample

Regression trees

$$\hat{\mathbb{E}}(y_i|X_i) = \sum_{k=1}^{K} \theta_k \mathbf{1}_{X_i \in C_k(L)} \qquad C_k(L) = \prod_{l=1}^{L} \mathbf{1}_{X_{i,j(l)} \gtrless \overline{X}_{j(l)}}$$

Forecast using the sample mean in a segment of the covariate space partition using a sequence of at most L cutoffs applied to one dimension at a time, for example:





Regression trees

$$\hat{\mathbb{E}}(y_i|X_i) = \sum_{k=1}^{K} \theta_k \mathbf{1}_{X_i \in C_k(L)} \qquad C_k(L) = \prod_{l=1}^{L} \mathbf{1}_{X_{i,j(l)} \gtrless \overline{X}_{j(l)}}$$

Forecast using the sample mean in a segment of the covariate space partition using a sequence of at most L cutoffs applied to one dimension at a time:

- Increasing L always increases in-sample fit, may lead to overfitting $\Rightarrow$ we usually penalize this or treat as a hyperparameter
- Can capture non-linear relations and interaction effects
- When applied to stocks, similar to multi-dimensional **portfolio sorts**, but instead of e.g. $3 \times 3 \times 3$ grids we use rectangles are of unequal size
- Similar flavor to **kernel regression** where you forecast using average of observations "close enough" to X_i
  - Regression lets data tell you what's close enough vs. imposing exogenously

Random forests

$$\hat{\mathbb{E}}(y_i|X_i) = \frac{1}{M}\sum_{m=1}^{M}\hat{\mathbb{E}}(y_i|Z_i(m)) \qquad \hat{\mathbb{E}}(y_i|Z_i(m)) = \sum_{k=1}^{K}\theta_k \mathbf{1}_{Z_i(m)\in C_{m,k}(L)},$$

where $Z_i(m)$ is one of M random smaller-dimensional subsets of X_i

- **Idea:** random trees sometimes overfit by finding tiny subsets with extreme values. Don't let them rely on any one element of X_i too much
- One of many **ensemble** methods that takes advantage of averaging forecasts across possible models
  - 'Wisdom of the model crowds'
- Results in conditional means that are smoother than a single tree
- Generally performs much better than a simple regression tree, one of the best plug-and-play methods because there are relatively few hyperparameters
- Can further improve, especially in economics settings, by allowing for local linearity instead of assuming conditional mean is locally constant

Boosting and gradient-boosted regression trees

$$\hat{\mathbb{E}}(y_i|X_i) = \sum_{m=1}^{M} f_m(X_i, \theta_m)$$

Boosting idea: instead of a single complex model $\mathbb{E}(y_i|X_i) = f(X_i, \theta)$ with parameters we estimate all at once, use a sequence of M simple models:

$$\begin{aligned}
\mathbb{E}(y_i|X_i) &= f_1(X_i, \theta) \\
\mathbb{E}(y_i - f_1(X_i, \theta)|X_i) &= f_2(X_i, \theta) \\
&\dots \\
\mathbb{E}(y_i - f_{M-1}(X_i, \theta)|X_i) &= f_M(X_i, \theta),
\end{aligned}$$

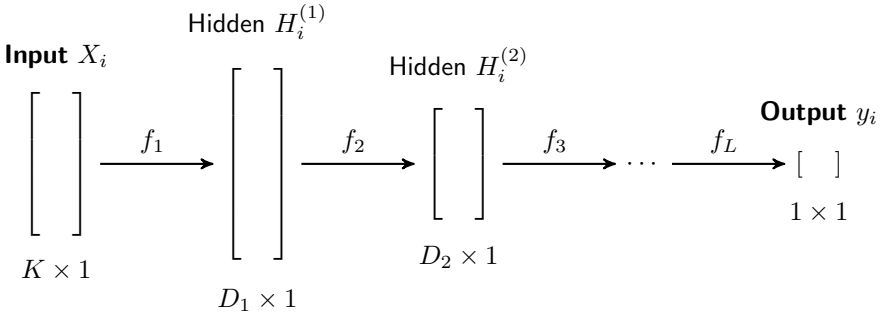
and form a final composite forecast

- Often used with simple trees with small L, called gradient-boosted trees

Neural networks: layers, hidden nodes, and activation functions

$$\hat{\mathbb{E}}(y_i|X_i) = (f_1 \circ f_2 \circ \cdots \circ f_L)(X_i, \theta)$$

Neural networks are composite functions that map inputs (X_i here) into outputs (Y_i here) using a sequence of operations (**layers**) loosely inspired by brain biology[1]



[1] I focus here on feed-forward neural networks, many other forms exist

Neural networks: layers, hidden nodes, and activation functions

$$\hat{\mathbb{E}}(y_i|X_i) = (f_1 \circ f_2 \circ \cdots \circ f_L)(X_i, \theta)$$

Neural networks are composite functions that map inputs (X_i here) into outputs (Y_i here) using a sequence of operations (**layers**) loosely inspired by brain biology[1]

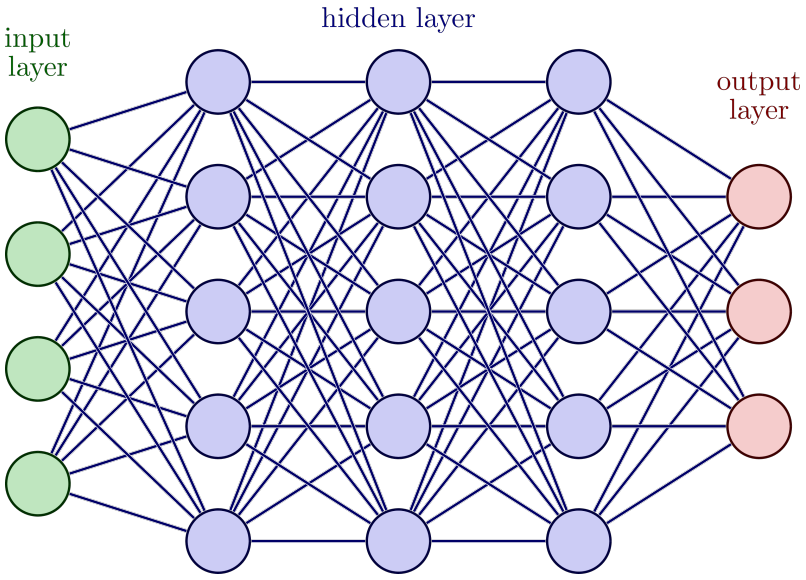
The functions f_l calculate each element (**node**) of a new layer as a linear combination of the nodes in the previous layer with a simple **activation function** applied:

$$H_i^{(l)}(j) = g\left(\theta_0^l(j) + \theta^{(l)}(j)' H_i^{(l-1)}\right),$$
$$g(z) = z\mathbf{1}_{z>0} \text{ (or similar function)}$$

where $\theta_0^l(j)$ is a "bias" or intercept term, and $\theta^{(l)}(j)$ is a $D_{l-1} \times 1$ vector of parameters (neurons?)

[1] I focus here on feed-forward neural networks, many other forms exist

Neural networks: layers, hidden nodes, and activation functions



Properties of neural networks

- **Multi-Layer Perceptrons (MLP)**: any subset of a network consisting of least three layers of nodes: an input layer, one or more hidden layers, and an output layer
  - Common setup is to have input -¿ higher-dimensional hidden -¿ output as one "MLP" stage inside a larger network
- **Universal approximation**: MLPs are powerful learning methods that, with enough nodes, can approximate virtually any continuous function with only a single hidden layer
- **Functional form flexibility**: NNs are explicitly designed to approximate complex nonlinear associations and interactions
- **Parameterization**: loads of parameters, one step in the composite function has $(D_{l-1}+1)D_l$ parameters
  - Over a million parameters common even for "simple" applications
- **GPU-friendly**: despite complexity and huge parameter space, mostly just matrix multiplication $(\theta^{(l)} \cdot H_i^{(l-1)})$, gradient easy to calculate
  - Modern GPUs optimized to do this very quickly

Shallow vs. Deep Learning

- **Shallow Networks**: fewer hidden layers (e.g., one hidden layer)
- **Deep Networks**: more hidden layers (e.g., three hidden layers or more)
- Despite NNs being universal approximators with only one hidden layer, in practice deeper networks typically perform better for the same parameter count
- In some asset pricing problems, "shallow" learning can outperform "deeper" learning, unlike in fields like computer vision
  - This is likely due to the comparative dearth of data and low signal-to-noise ratio in asset pricing problems
  - Neural network performance for individual stock returns often peaks at a moderate number of layers (e.g., three hidden layers) and then declines

Embedding and Autoencoding

Embedding is a map from a high-dimensional, complex input to a lower-dimensional vector output that retains as much information as possible

- For simple numeric inputs, this is a form of dimension reduction
- For categorical, text, or image inputs this is a more-complex mapping
  - Classic example is word embeddings, e.g. "Word2Vec", which represents each word as a vector with e.g. 300 dimensions. Gives you cool things like:

$$\overrightarrow{\text{Sushi}} - \overrightarrow{\text{Japan}} + \overrightarrow{\text{Germany}} \approx \overrightarrow{\text{Bratwurst}}$$

An **autoencoder** learns a lower-dimensional representation of the input vector (form of dimension reduction)

- Trained along side a "decoder" that restores the original dimensionality, try to match as close as we can

BREAK

ML in empirical asset pricing: possible applications

Stock return predictability

Canonical measurement problem in asset pricing: estimate

$$\mathbb{E}(r_{i,t+1}|I_t),$$

where $r_{i,t+1}$ is the excess return of asset i and I_t is the unobservable information set of investors at time t

- Once measured, can work on understanding its economic determinants

Gu, Kelly, and Xiu (2019) show:

1. Allowing for non-linearities and interaction effects matters
2. Value added for machine learning is larger for predicting market/portfolio returns in the time-series than it is for predicting cross-section
3. Economic gains from machine learning forecasts are large
   - 0.77 out-of-sample Sharpe Ratio for market timing strategy, 1.35 for value-weighted long-short strategy
4. Best predictors are price trends, liquidity, and volatility variables
5. Strategies have high turnover, overweight microcaps $\Rightarrow$ sensitive to txn costs

Bond return predictability

Bianchi, Büchner, and Tamoni (2021): can apply similar methods to measure conditional bond risk premia

- Topic of Jay's presentation today

Textual data in asset pricing: narrative factors

Vast amounts of textual data produces for publicly-traded firms

- Earnings conference call transcripts (e.g. Barth, Mansouri, and Woebbeking (2023))
- 10-K and 10-Q filings contain length descriptions in addition to the raw accounting numbers (e.g. Frankel, Jennings, and Lee (2021))
- News coverage, press releases, product announcements, legal filings, etc.

Question: can we learn about risk exposure, risk premia, sentiment, etc. from these?

Bybee, Kelly, and Su (2023):

- Integrates topic modeling ("LDA") to group terms from WSJ into interpretable narrative themes (e.g., "Recession," "Economic Growth")
- Uses "Sparse IPCA" (more on this later in the semester) to map narrative attention series into a small number of common asset pricing risk factors, selecting relevant narratives with a Lasso penalty
- Narrative factors achieve higher out-of-sample Sharpe ratios and smaller pricing errors than standard characteristic-based models

Market microstructure: predicting market dynamics

Question: does the 'plumbing' – frictions in the trading process that create illiquidity – matter in asset pricing?

- One test: do empirical liquidity measures predict changes in market conditions (volatility, return autocorrelation, etc.)?

Easley, López de Prado, O'Hara, and Zhang (2021):

- Use a random forest to predict important market price dynamics (e.g., changes in bid-ask spread, volatility, skewness, kurtosis)
- **Features**: standard empirical microstructure measures (e.g., Roll measure, Kyle's λ, Amihud measure, VPIN, VIX – see Liquidity lecture) from tick data
- **Within-Asset Predictions**: For a small number of own-asset features, random forests and logistic regression show similar predictive power, but random forests excel when many features are considered
- **Cross-Asset Effects**: Including cross-asset features (e.g., from commodity, equity, currency, and fixed-income futures) significantly enhances predictive power
  - Hints at systemic components of liquidity shocks

Mutual fund skill prediction

Summary of mutual fund performance literature (we detail later): average active equity fund **underperforms** market after fees, **no persistence** in performance year to year

Kaniel, Lin, Pelger, and Van Nieuwerburgh (2023) shows that fund characteristics can consistently forecast which mutual funds will outperform using ML

- **Method**: neural network, one hidden layer with 64 nodes leveraging lagged fund-specific and macroeconomic characteristics [130, 131].
- Fund momentum and fund flow are the most important predictors of future risk-adjusted fund performance
- They uncover novel and substantial **interaction effects** between sentiment and both fund flow and fund momentum
- Outperformance persists for more than three years, and returns of predictive long-short portfolios are higher following periods of high sentiment
  - Typically can't short mutual funds, so "long-short" is a purely statistical comparison

Machine learning strengths and weaknesses

Strengths of machine learning:

- Out-of-sample fit
- Interaction effects and non-linearity
- Can have large number of RHS variables

Weaknesses of machine learning:

- Focus on prediction/forecasting problem, nothing to say about causation or hypothesis testing
  - Newer methods estimate causal average treatment effect (CATE), e.g. Hahn, Murray, and Carvalho (2020)
- Currently black-box, hard to interpret
- Too many degrees of freedom for researcher
  - Many papers mitigate this by showing 10-15 different ML implementations, leaving readers unclear what the right approach is and how to handle disagreement
- Requires lots of data

What is AI and how does it relate to ML?

Lots of definitions of "AI" going back decades, but current revolution rests on a few core recent innovations:

1. "Transformer" architecture
2. Vast computing resources
3. Vast training data sets

to produce something that feels closer to a **general intelligence** than something that can work only on narrow tasks (like Chess or Go AI)

AI uses the tools of machine learning to accomplish a broader task

How LLM chat-bots work

Intuition: train a massive neural network to predict the next word in any sequence of text, then tell it to predict the next word in a "script" that looks like:

> *What follows is a conversation between a user and helpful, very knowledgeable AI assistant. The AI assistant never suggests violence and loves pizza.*
>
> User: Give me a recipe for dinner using avocado, salmon, carrots, and basic staples.
>
> AI Assistant: Here you _____

The initial instruction is called the **system prompt** and is extremely important to the user experience

- Can be partially customized in both the web apps and API
- Much more complicated than the above in practice

Everything before the next word we need to predict is called the **prompt** or **context**

How LLM chat-bots do NOT work

- No massive sequence "if/then" statements covering all possible cases
- No knowledge outside of the "Context" given and whatever it learns in training
  - Doesn't know any news past when training data ended (**knowledge cutoff**)
  - Doesn't know where you work, who your friends/family are, etc. unless you tell them – unless you're famous, it's not worth memorizing all this stuff
  - Context lengths are limited (200k words is order of mangitude), so LLMs forget things you told them earlier
- No "in the moment" Google search
  - Tools that do this, such as Perplexity, combine Google and LLM APIs to feed Google search results into an LLM for summary
    - This type of thing is called **scaffolding**
- No objectives other than predicting what they helpful AI assistant described in the system prompt would say, getting the "thumbs up" reward, and avoiding the "thumbs down" reward
  - The network parameters are tuned after initial training using **reinforcement learning from human feedback** (RLHF)

LLM data flow

When you prompt an LLM, the following steps occur

1. **Tokenize** the entire prompt, breaking words, numbers, code, symbols into one of ∼100k distinct **tokens**

2. Map each token to a learned **embedding** E_i – a high-dimensional (order of 25k) vector – that conveys its meaning in isolation
3. Let all the embeddings "talk" to each other using the attention mechanism, so we can figure out that "stream" means something different in "we swam in the cold stream" and "I watched a live stream of the game"
4. Update each embedding using an MLP
5. Repeat 4-5 hundreds of times
6. Use a learned mapping to go from the final embedding in the context to our guess at the next token

“Attention Is All You Need” (NIPS 2017)

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

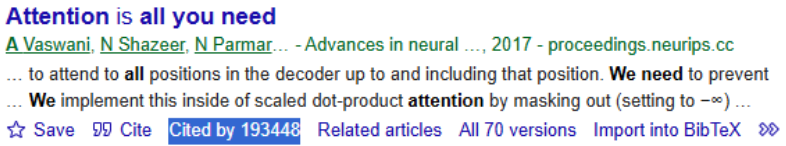
Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

“Attention Is All You Need” (NIPS 2017)



For comparison, Jensen and Meckling (1976) has 145k, Fama French (1993) has 39k, Sheridan has 105k, Laura has 40k

“Attention Is All You Need” (NIPS 2017)

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

"Attention Is All You Need" (NIPS 2017)

New **Transformer** architecture outperforms frontier translation models despite only four days of training on 8 GPUs

- A major advance in ML for translation, but also enabled nearly all generative AI

Attention mechanism: update embedding $\overrightarrow{E}_i$ using messages:

$$E_i' = E_i + \sum_j M_{ji}, \qquad M_{ji} = \underbrace{\underbrace{\underbrace{\text{softmax}}_{\text{normalization}}\left(W_{\text{key}}E_j \cdot W_{\text{query}}E_i\right)}_{\text{Attention weight}\in[0,1]}\underbrace{W_{\text{value}}E_j}_{\text{Value}}}$$

- Allows any token to 'talk to' any other token
- Parallelizable because it's all matrix multiplication & final step is additive
  - Can even do multiple **attention heads** at once (many distinct combinations of $W_{\text{key}}, W_{\text{query}}, W_{\text{value}}$)

AI in the research process

We all need to make ourselves complements to AI

- Humanity's great comparative advantage is flexibility
- Move from "Man vs. Machine" to "Man + Machine"
- **Centaur Analysts**: Just as in chess, combining human and AI capabilities ("centaur analysts") can lead to superior performance compared to either alone
- Assign AI the sub-tasks you are weakest at, enjoy the least, or add the least value to (**comparative advantage** idea)

Goal is to dramatically increase output – for us, quality and quality of research

- Treat AI as a tool to master

AI in the research process

Possible use cases for AI:

- **Data Processing and Cleaning**: AI excels at digesting large volumes of information**, including unstructured textual data from SEC filings, news, and social media
- **Coding**: AI can assist with code generation, debugging, and optimization, enhancing efficiency
- **Idea Development**: AI can be a sounding board, 'bad idea' generator, brainstorming partner, and on-demand feedback provider
- **Literature Review**: AI can rapidly process and summarize vast amounts of literature, identifying key themes, methodologies, and gaps
- **Communication**: generating beautiful diagrams, figures, presentations, illustrations, has never been easier

AI in research itself

Leveraging foundational model APIs

- Researchers can query and leverage pre-trained large language models or other foundational AI models through APIs to extract insights, generate text, or perform complex analytical tasks relevant to finance

Training New "AI" Models Tailored to Finance Tasks:

- **AI forecasters**: Researchers can build their own AI models, trained on timely, publicly available data (firm-level, industry-level, macroeconomic, textual data) to predict financial outcomes like stock returns or earnings
- **Alternative Data**: AI is essential for processing new classes of "alternative data" (e.g., stock price history images, satellite images of parking lots, social media) that provide unique and timely clues for investment opportunities
- **Integrating Economic Theory**: Incorporating economic restrictions or structural insights directly into ML algorithm design (e.g., for causal inference, consumer choice, or panel data structures)

Discussion