Data Science for Economists

Introduction to Machine Learning

Mox Ballo

University of the Philippines

April 23, 2025

SCHOOL OF ECONOMICS
UNIVERSITY OF THE PHILIPPINES DILIMAN, QUEZON CITY

Overview

- Learning = improving performance at a task through experience
- Formal set-up: input x, output y, unknown target function f
- Goal: learn an approximation $g(\mathrm{x})$ using data
- Example: Forecasting electricity demand from past weather and consumption

Set-up of the Learning Problem

- **Inputs:** $\mathrm{x} \in \mathcal{X}$ — observed variables (e.g., hour, weather, previous demand)
- **Outputs:** $y \in \mathcal{Y}$ — quantity to predict (e.g., next-hour electricity load)
- **Input Space:** $\mathcal{X}$ — all possible combinations of input features
- **Output Space:** $\mathcal{Y}$ — all valid values the output can take
- **Unknown Target Function:** $f : \mathcal{X} \to \mathcal{Y}$ — true, hidden mapping
- **Learning Objective:** Find an approximation $g \in \mathcal{H}$ such that $g(\mathrm{x}) \approx f(\mathrm{x})$

Learning Problems in Economics

- **Demand forecasting:** Predict electricity load based on time, weather, and historical usage
- **Price elasticity estimation:** Learn how quantity demanded responds to price changes using observed market data
- **Consumer choice modeling:** Predict product choice given consumer attributes and product features

Input Space and Feature Selection

Problem: Predict hourly electricity demand to optimize production.

- **Vector of Inputs** (x_n): hour of day, day of week, temperature, humidity, historical consumption
- **Input Space** ($\mathcal{X}$):

$$\mathcal{X} = \{0\text{-}23\} \times \{\text{Mon-Sun}\} \times \mathbb{R}^{+} \times [0, 100] \times \mathbb{R}^{+}$$

Output and Hypothesis Space

- **Output (y_n):** The observed electricity demand, e.g. in megawatts (MW), at time n.
- **Output Space ($\mathcal{Y}$):** Since demand cannot be negative, we model outputs in the space of non-negative real numbers: $\mathcal{Y} = \mathbb{R}^+$.
- **Hypothesis Space ($\mathcal{H}$):** The set of functions $g : \mathcal{X} \to \mathcal{Y}$ that map input features (e.g., time of day, weather) to a predicted demand value.
  - Linear models: $\mathcal{H} = \{g(x) = \mathrm{w}^\top x\}$
  - Polynomial regression: $\mathcal{H}$ includes higher-order terms to capture nonlinear trends.
  - Nonparametric models: GAMs or decision trees allow more flexible hypothesis spaces.

Learning Algorithm

- **Goal:** Use training data $\mathcal{D} = \{(\mathrm{x}_n, y_n)\}_{n=1}^{N}$ to choose a hypothesis $g \in \mathcal{H}$ that approximates the unknown target function f.
- **Learning Algorithm ($\mathcal{A}$):** A systematic procedure that selects the final hypothesis:

$$g = \mathcal{A}(\mathcal{D}) \in \mathcal{H}$$

- **Examples for electricity demand:**
  - Linear regression (OLS, Ridge, Lasso)
  - Tree-based models (Random Forest, XGBoost)
  - Neural networks for time series
- **Output:** A learned function g that generalizes well to new inputs x_{new}

Error Measures

- **Final Hypothesis:** The learned function $g(\mathrm{x}) = f(\mathrm{x}; \hat{\beta})$ aims to approximate the unknown target f.
- **In-Sample Error (Training Error):**

$$E_{\text{in}}(g) = \frac{1}{N} \sum_{n=1}^{N} (g(\mathrm{x}_n) - y_n)^2$$

  Measures how well g fits the training data.
- **Out-of-Sample Error (Generalization Error):**

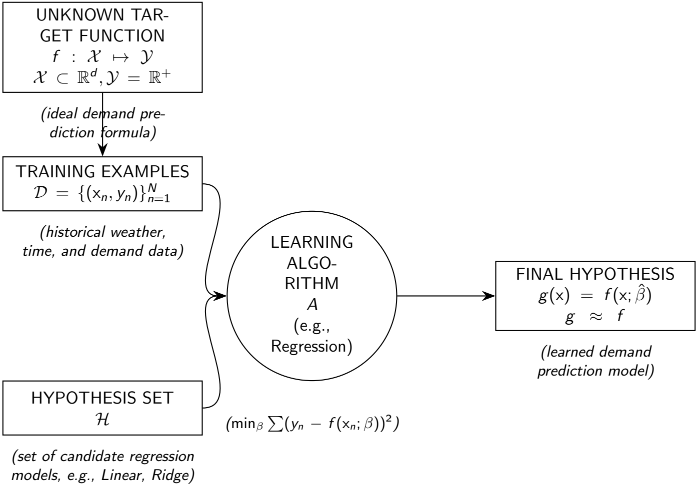
$$E_{\text{out}}(g) = \mathbb{E}_{\mathrm{x},y}\left[(g(\mathrm{x}) - y)^2\right]$$

  Measures how well g performs on new, unseen data.

Error Measures

- **Goal of Learning:** Minimize $E_{\text{out}}(g)$, not just $E_{\text{in}}(g)$. This ensures good generalization.
- **Common Metrics:**
  - **MSE (Mean Squared Error)** — most common for regression.
  - **MAE (Mean Absolute Error)** — less sensitive to outliers.
  - **MAPE (Mean Absolute Percentage Error)** — often used in electricity demand forecasting.
- **Why it matters:** A low in-sample error doesn't guarantee a low out-of-sample error — hence the need for validation techniques (e.g. cross-validation).

Basic Set-up: Electricity Demand Prediction



A Simple Learning Model

- Consider forecasting the electricity demand for a given hour in a day, given weather and time-based conditions.
- The training examples $\mathcal{D} = \{(\mathrm{x}_n, y_n)\}_{n=1}^{N}$ come from historical demand logs.
- We choose a hypothesis set $\mathcal{H}$, and a learning algorithm $\mathcal{A}$, to find a function $g \approx f$ that maps inputs x to predicted demand y.
- A simple model: assume demand can be predicted as a weighted sum of features:

$$g(\mathrm{x}) = \sum_{i=1}^{d} w_i x_i + b$$

- for simplicity, instead of β, we will use w

What's in the Input Vector x?

- $\mathrm{x} \in \mathbb{R}^d$, where each component of x represents:
  - x_1: Hour of the day (0–23)
  - x_2: Day of the week
  - x_3: Temperature
  - x_4: Humidity
  - x_5: Lagged demand (e.g. demand one hour ago)
- Output $y \in \mathbb{R}^+$: actual electricity demand in megawatts (MW)
- Our goal: learn a function $g(\mathrm{x})$ that gives accurate predictions of y

Linear Hypothesis for Demand

- We define our hypothesis space $\mathcal{H}$ to consist of linear models:

$$h(\mathrm{x}) = \sum_{i=1}^{d} w_i x_i + b$$

- w_i: learned importance of each input feature (e.g., temperature, time)
- b: bias term, reflects baseline electricity demand
- Compact form:

$$h(\mathrm{x}) = \mathrm{w}^\top \mathrm{x} + b$$

- After training, we get final hypothesis:

$$g(\mathrm{x}) = h(\mathrm{x}; \hat{\mathrm{w}})$$

Learning the Parameters: ERM

- We use the training data to estimate weights $\hat{\mathrm{w}}$ and bias $\hat{b}$
- Objective: minimize squared error (Empirical Risk Minimization)

$$\min_{\mathrm{w},b} \frac{1}{N} \sum_{n=1}^{N} \Big(y_n - (\mathrm{w}^\top \mathrm{x}_n + b) \Big)^2$$

- This selects the best-fitting line (or hyperplane) in the training data
- The result is our final hypothesis:

$$g(\mathrm{x}) = \mathrm{w}^\top \mathrm{x} + b$$

Learning vs. Design

- In econometrics, models are often specified from theory: *design-based*.
- In machine learning, we let the data suggest the best hypothesis: *data-driven*.
- Tradeoff:
  - Design-based: interpretable, theory-aligned, but may misrepresent data.
  - Learning-based: better predictive accuracy, but harder to interpret and justify causally.
- Our goal is to understand when and how data-driven models (like this) can augment economic analysis.

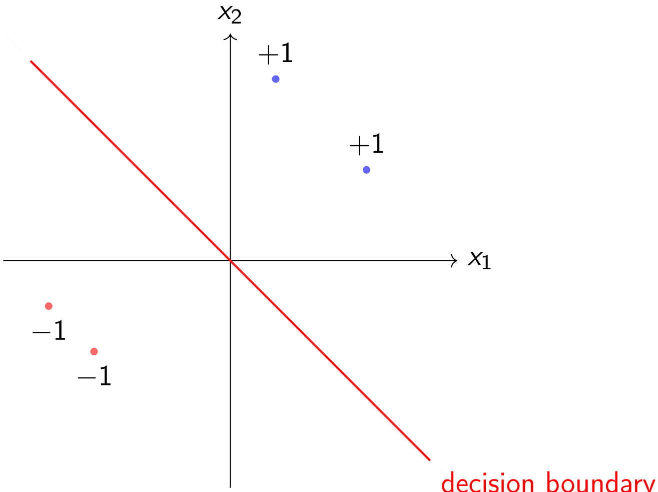
Digression: Perceptrons

- A perceptron is a simple binary classifier.
- It takes multiple inputs (features), computes a weighted sum, and applies a decision rule:

$$h(\mathrm{x}) = \mathrm{sign}(\mathrm{w}^\top \mathrm{x} + b)$$

- Think of it as a "yes/no" decision boundary — like deciding if the grid is stable or at risk.
- Inspired by how neurons in the brain work: inputs fire, and if the signal is strong enough, there's an activation (output = 1).

Illustration: 2-Feature Case



Digression Summary

- A perceptron is one of the simplest machine learning models.
- It separates the space into two regions using a line (or hyperplane in higher dimensions).
- It's fast and interpretable — great for simple decisions like:
  - Is the power grid at risk of overload?
  - Should we activate a reserve power plant?
- But it only works well if the data can be separated by a straight line (linearly separable).

Perceptron-Style Binary Classification

- In some energy markets, the goal is to make a **binary decision**:

$$y \in \{-1, +1\} \quad \text{(e.g., brownout risk?)}$$

- Input vector x might include:
  - x_1: Forecasted demand (MW)
  - x_2: Available generation capacity
  - x_3: Grid frequency deviation
  - x_4: Weather forecast (e.g., typhoon alert)
- Output: $y = +1$ means "risk of brownout" and $y = -1$ means "normal operation"

Perceptron Model

- Define hypothesis:

$$h(\mathrm{x}) = \mathrm{sign}(\mathrm{w}^\top \mathrm{x} + b)$$

- Interpretation:
  - If $h(\mathrm{x}) = +1$: trigger preventive measures (e.g., activate reserve plants)
  - If $h(\mathrm{x}) = -1$: system is stable
- The learning algorithm finds weights w that separate high-risk vs. low-risk conditions.
- If input space is linearly separable, the Perceptron algorithm converges to a solution.

Philippine Examples of Binary Energy Decisions

- **Scenario 1: Load Shedding**
  - NGCP needs to decide whether to issue yellow or red alerts
  - Predict based on forecast demand and supply margin
- **Scenario 2: Rotational Brownouts**
  - During the dry season, hydroelectric capacity is low
  - Predict risk of brownout in provinces like Palawan or Mindoro
- **Scenario 3: Distributed Energy Dispatch**
  - If risk $h(\mathrm{x}) = +1$, then dispatch diesel backup generators in off-grid areas

Supervised Learning

- In supervised learning, the algorithm is given input-output pairs (x, y).
- The goal is to learn a function $g(\mathrm{x})$ that predicts y from x.
- Two major types:
  - **Regression:** Predict continuous output (e.g. electricity demand in MW)
  - **Classification:** Predict binary or categorical output (e.g. default vs. no default)
- The algorithm is "supervised" by the correct answer.
- **Examples:**
  - Predicting household electricity consumption based on time and weather
  - Forecasting inflation using macroeconomic indicators
  - Predicting approval for microfinance loans based on applicant data

Active Learning in Supervised Leadning)

- In standard supervised learning, the algorithm passively receives labeled data.
- In **active learning**, the learner can query for labels — it chooses which x to ask for a y.
- Motivation: labels may be expensive or time-consuming to obtain.
- **Example:** Smart meters collect hourly consumption (x), but manual surveys provide income class (y).
- Active learning asks: "Which households should we label (survey) to improve our model fastest?"
- **Examples:**
  - Reducing survey costs in poverty targeting
  - Sampling households for tariff adjustment studies
  - Choosing which firms to audit for competition enforcement

Online Learning

- In **online learning**, the algorithm updates its model one observation at a time.
- Useful for streaming data or when retraining on full data is costly.
- At each step t, the learner:
  1. Gets input x_t
  2. Predicts $\hat{y}_t = g_t(\mathrm{x}_t)$
  3. Receives actual y_t and incurs loss
  4. Updates model to g_{t+1}
- **Examples:**
  - Real-time pricing forecasts in electricity markets (e.g., WESM)
  - Updating inflation predictions with streaming macroeconomic indicators
  - Learning user behavior in mobile payment platforms

Unsupervised Learning

- In unsupervised learning, we are given only inputs $x_1, x_2, \ldots, x_N$, with no labels or targets.
- objective: discover patterns, structures, or relationships within the data.
- The algorithm is "unsupervised" — it must learn without knowing the correct answer.
- Useful when labels are unavailable or ambiguous.
- **Main types:**
  - **Clustering** (e.g. k-means, hierarchical clustering)
  - **Dimensionality reduction** (e.g. PCA, t-SNE)

Clustering

- Clustering groups similar observations together based on a distance or similarity metric.
- Examples:
  - K-means clustering
  - Hierarchical clustering
  - DBSCAN
- Clustering reveals hidden groupings in the data — no labels are needed.
- **Examples:**
  - Segmenting consumers based on electricity usage patterns
  - Grouping municipalities by poverty and infrastructure indicators
  - Identifying similar firms in competition analysis

Dimensionality Reduction

- Reduces the number of features while retaining the most important variation in the data.
- Techniques:
  - Principal Component Analysis (PCA)
  - t-SNE for visualization
  - Autoencoders (neural networks)
- Helps:
  - Visualize high-dimensional data
  - Remove noise or redundancy
  - Improve downstream supervised learning
- **Example:**
  - Reducing 50 household expenditure categories into 2–3 latent "lifestyle" dimensions

Supervised vs. Unsupervised Learning

Aspect	Supervised Learning	Unsupervised Learning
Input	Features + labels (x, y)	Features only (x)
Goal	Predict labels	Find structure
Feedback	Known correct outputs	No direct feedback
Methods	Regression, classification	Clustering, PCA

Reinforcement Learning

- In reinforcement learning, an agent learns by interacting with an environment.
- It receives feedback in the form of rewards and learns a policy to maximize long-term reward.
- No fixed training set — learning happens over time via trial and error.
- **Examples:**
  - Dynamic pricing of electricity under real-time market conditions
  - Adaptive congestion pricing in transport networks
  - Policy simulation: learning optimal subsidy schemes under changing economic conditions
- Especially useful for problems involving time, strategy, and sequential decisions.

Key Components of Reinforcement Learning

- **Agent:** Learner or decision-maker (e.g., National Grid operator)
- **Environment:** World with which the agent interacts (e.g., electricity grid)
- **State (s):** Current situation observed by the agent (e.g., demand level, reserve margin)
- **Action (a):** Decision the agent can take (e.g., dispatch reserve, adjust price)
- **Reward (r):** Feedback signal after taking action (e.g., blackout avoided $= +10$; blackout $= -100$)
- **Policy (π):** Mapping from state to action: $a = \pi(s)$
- **Goal:** Learn the best policy π^* that maximizes long-term cumulative reward

What is the "Training Set" in RL?

- Unlike supervised learning, RL does not receive labeled pairs (x, y).
- Instead, the agent observes a sequence of experience tuples:

$$(s_t, a_t, r_t, s_{t+1})$$

- This tells the agent:
  - "I was in state s_t"
  - "I chose action a_t"
  - "I received reward r_t"
  - "The new state is s_{t+1}"
- Learning happens by updating value estimates or policies from this feedback.

Where is RL Useful?

- Reinforcement Learning is ideal when:
  - The agent must make sequential decisions
  - The environment changes dynamically
  - Outcomes depend on both present and past decisions
- **Possible Applications:**
  - Energy markets: dispatch decisions, real-time pricing
  - Transport economics: dynamic tolling, congestion response
  - Social policy: testing subsidy levels under uncertain behaviors
  - Public health: optimizing allocation of limited vaccines over time
- RL is often used in simulation-based environments when real-world experimentation is costly.

Is Learning Always Feasible?

- Learning is only useful if we can generalize from the data.
- Key question: If a model works well on training data, will it also work on new, unseen data?
- This is the difference between:
  - **Memorization** — fitting the training data exactly
  - **Generalization** — capturing the true underlying pattern
- Learning is feasible when we can ensure small error on unseen data (out-of-sample).
- This requires assumptions on:
  - How the data is generated
  - The complexity of the hypothesis space

Condition 1: How the Data is Generated

- We assume the training data $(x_1, y_1), \ldots, (x_N, y_N)$ are drawn independently from the same probability distribution $\mathcal{D}$.
- This is the i.i.d. assumption (independent and identically distributed).
- Why it matters:
  - If data changes over time (non-stationary), past experience may not generalize.
  - If observations are correlated (e.g., clustered or networked), standard learning guarantees break down.
- **Examples:**
  - Forecasting electricity demand works if usage patterns are stable — not during blackouts or crises.
  - Learning a pricing model for MC Taxi fares only works if commuter behavior is consistent.
- Takeaway: For learning to generalize, the future must "look like" the past.

Condition 2: Complexity of the Hypothesis Space

- A hypothesis space $\mathcal{H}$ is the set of functions we allow the learner to choose from.
- If $\mathcal{H}$ is too rich (too many flexible models), it can perfectly fit the training data but fail to generalize.
- The more complex the hypothesis set, the more data we need to ensure learning is feasible.
- Complexity can be measured by:
  - VC Dimension
  - Number of parameters
  - Model structure (e.g., depth of trees, layers of a neural net)
- **Examples:**
  - A flexible forecasting model with dozens of lag terms, interaction effects, and splines may overfit if sample size is small.
  - Simpler models (e.g., linear regression with 3–4 predictors) are more robust in small datasets.
- Trade-off: More expressive models vs. risk of overfitting.

Why Probability Matters in Learning

- Learning is about drawing conclusions from finite samples.
- Even if we fit the training data well, we might just be overfitting — capturing noise.
- We use probability to:
  - Estimate the likelihood of making a mistake on unseen data
  - Bound the difference between in-sample error and out-of-sample error
- **Example:** If we train a demand forecasting model on dry-season data, can it generalize to rainy season?
- Formal tools: concentration inequalities, generalization bounds

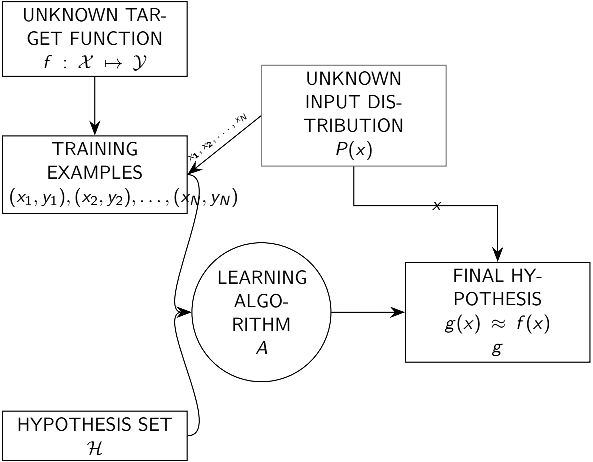
Putting It All Together

- Learning is feasible when:
  1. The data is i.i.d. from a fixed distribution
  2. The hypothesis space has controlled complexity
- We can then bound the difference between:

$$E_{\text{out}}(g) \approx E_{\text{in}}(g)$$

- This is the heart of generalization.
- In economics and policy:
  - Make sure your training data is relevant to your policy domain.
  - Don't fit overly flexible models without enough observations.

Learning Setup with Probability



The VC Dimension

- The **VC Dimension** (Vapnik–Chervonenkis) measures the complexity of a hypothesis set.
- Informally: it's the largest number of points that can be "shattered" (i.e., labeled in all possible ways).
- **Higher VC dimension** $\rightarrow$ more expressive models, but also higher risk of overfitting.
- **Lower VC dimension** $\rightarrow$ simpler models, may underfit.
- Goal: find a balance where the hypothesis set is expressive enough but still generalizes well.

Examples of VC Dimension

- **Threshold function on the real line:** VC dimension $= 1$
- **Linear classifiers in 2D:** VC dimension $= 3$
- **Perceptron with d inputs:** VC dimension $= d + 1$
- **Implication for electricity demand prediction:**
  - If you use too many features (hour, temperature, weather alerts, device types, etc.), you may overfit.
  - VC dimension helps guide the size of the hypothesis space.

Summary: When is Learning Feasible?

- Learning is feasible when:
  - The hypothesis space is not too large
  - The number of training examples is sufficient
  - The data distribution is stable over time
- Probability theory helps us quantify generalization error.
- VC dimension tells us when a model is too complex.
- **In economics:** Learning is feasible when:
  - You don't have too many instruments relative to observations
  - Consumer behavior is relatively stable

Understanding Error in Learning

- In supervised learning, we want our hypothesis g to approximate the true function f.
- But even the best model g will make some mistakes.
- We distinguish between:
  - **In-sample error** E_{in}: How well g performs on training data
  - **Out-of-sample error** E_{out}: How well g performs on new, unseen data
- Goal: minimize E_{out}, not just memorize training data.

Common Error Measures

- **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}(g(\mathrm{x}_n) - y_n)^2$$

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{N}\sum_{n=1}^{N}|g(\mathrm{x}_n) - y_n|$$

- **Binary Classification Error:**

$$\text{Error Rate} = \frac{1}{N}\sum_{n=1}^{N}1\left[g(\mathrm{x}_n) \neq y_n\right]$$

- Choice depends on the task and whether large errors should be penalized more.

What is Noise?

- **Noise** refers to the randomness or uncertainty in the relationship between x and y.
- Two data points with identical x may have different y values.

$$y = f(\mathrm{x}) + \text{noise}$$

- Noise is often due to:
  - Measurement errors
  - Omitted variables
  - Human unpredictability
- Noise sets a lower bound on the best achievable performance.
- Even the best model can't predict noise — only patterns.

Noisy Targets in Economics

- Many economic outcomes are inherently noisy:
  - Electricity consumption depends on mood, appliance use, etc.
  - Household income may fluctuate due to informal jobs
  - Self-reported survey data often contains inaccuracies
- As a result:
  - Don't expect perfect accuracy — measure error realistically
  - Focus on robustness and generalization, not fitting every detail
- **Design implication:** Simpler models may outperform complex ones in noisy settings.

Summary:

- Total error can be broken down into:

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

- **Bias:** Error due to wrong assumptions (underfitting)
- **Variance:** Error due to model sensitivity to data (overfitting)
- **Noise:** Irreducible error from randomness in the data
- As economists and data scientists, our job is to minimize bias and variance — not noise.

Bias-Variance Tradeoff

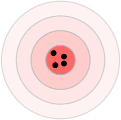
- When we train a model, the goal is to predict well — not just memorize.
- But prediction error has **three ingredients**:

$$\text{Total Error} = \underbrace{\text{Bias}^2}_{\text{wrong model}} + \underbrace{\text{Variance}}_{\text{too flexible}} + \underbrace{\text{Noise}}_{\text{randomness}}$$

- **Bias** is the error from using a model that's too simple (e.g., a straight line for something curvy).
- **Variance** is the error from using a model that's too complex — it overreacts to small changes in the data.
- **Noise** is unavoidable randomness (e.g., people leaving the aircon on by accident).
- **Tradeoff:**
  - A simple model (low variance, high bias) may miss real trends.
  - A complex model (low bias, high variance) may chase noise.
- We want a balance — not too simple, not too wiggly.

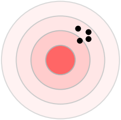
Visualizing Bias vs. Variance

Low Bias,
Low Variance



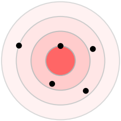
Accurate & consistent

High Bias,
Low Variance



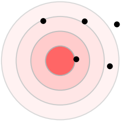
Inaccurate, consistent

Low Bias,
High Variance



Accurate (avg),
inconsistent

High Bias,
High Variance



Inaccurate & inconsistent