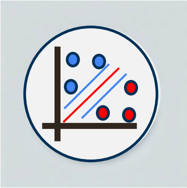
# Algorithms Introduction

**algorithm** (**al**-ge-ri-thəm) n. *a procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation*

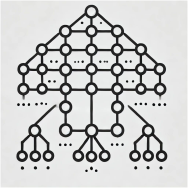
# Key ML Algorithms

- Bias vs Variance
- Complexity – Interpretability – Accuracy
- Data availability
- Training time and cost
- Scalability
- Performance
- Generalization vs Specialization
- Robust vs Flexibility

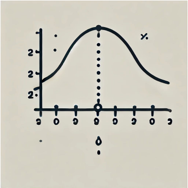
Regression



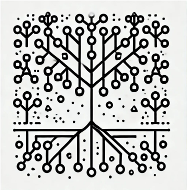
SVM



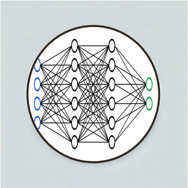
Decision Tree



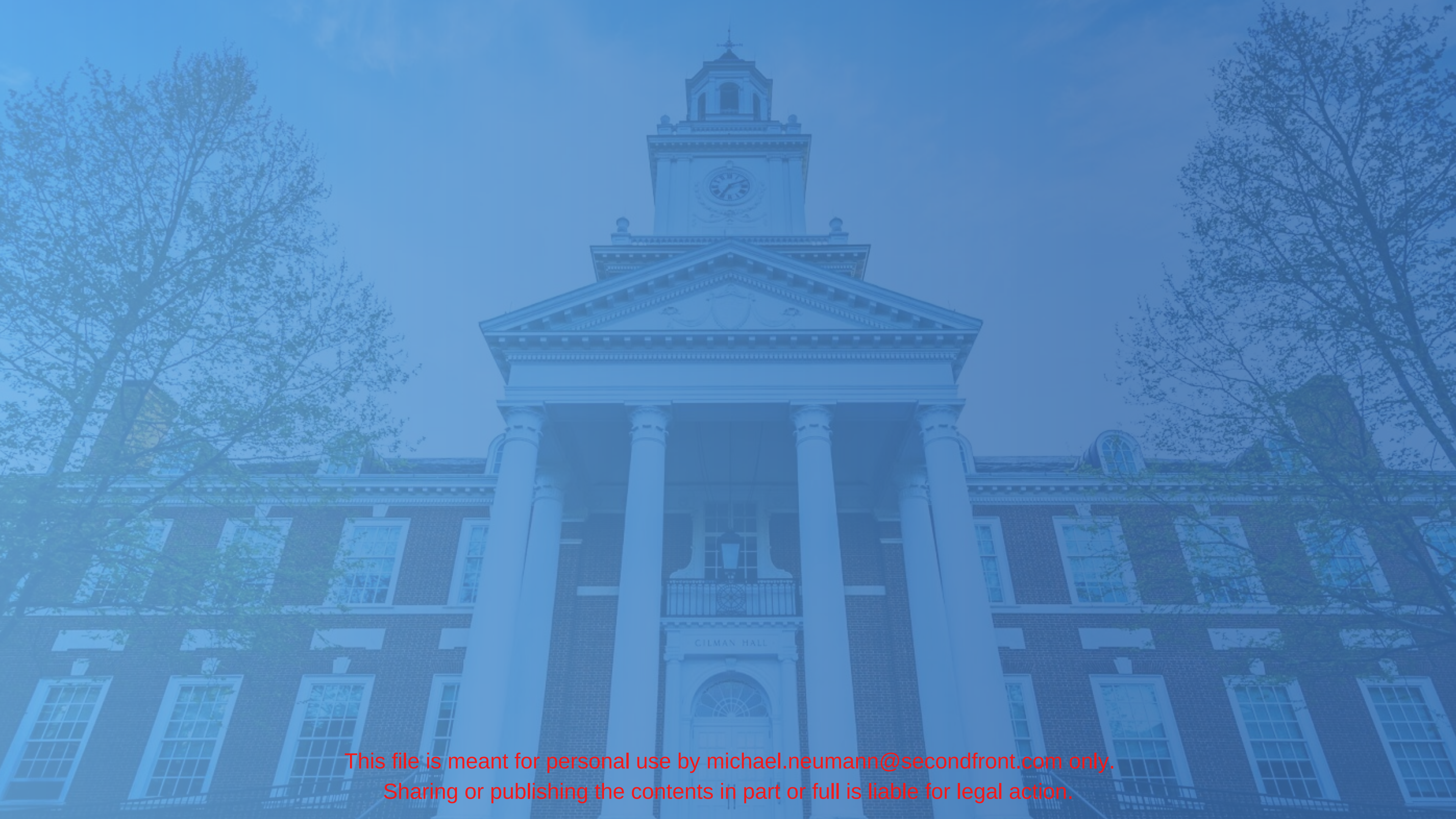
Naïve Bayes



Random Forest

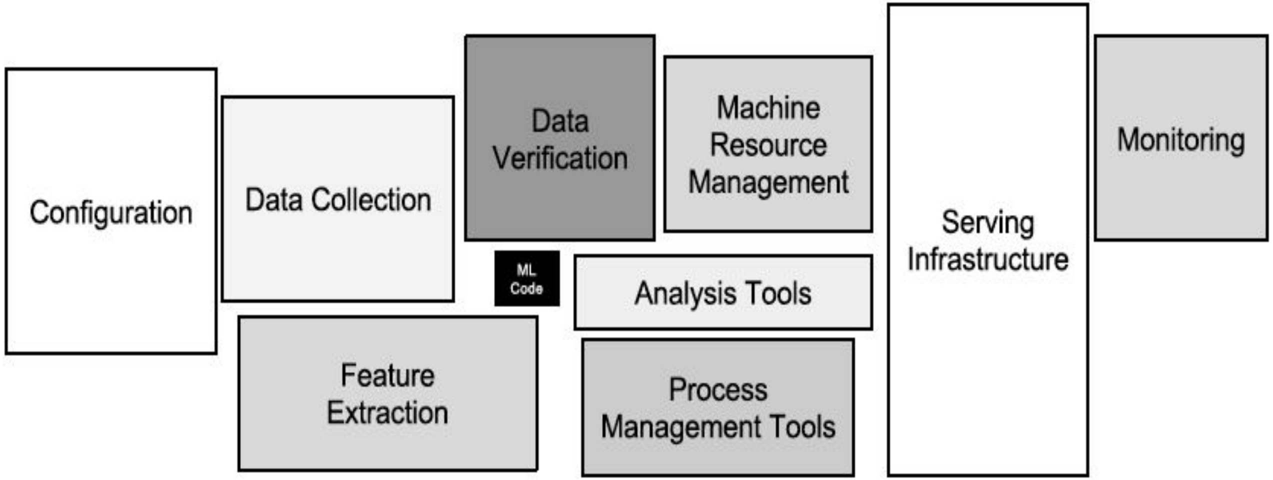


Neural Network



# AI Lifecycle

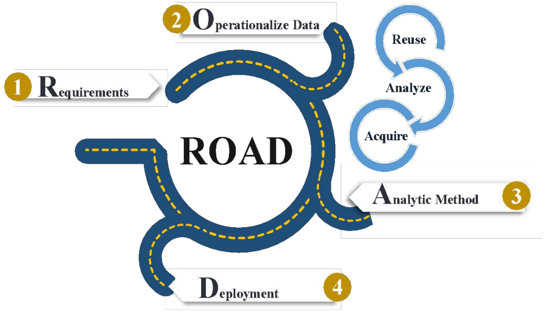
# AI within Context

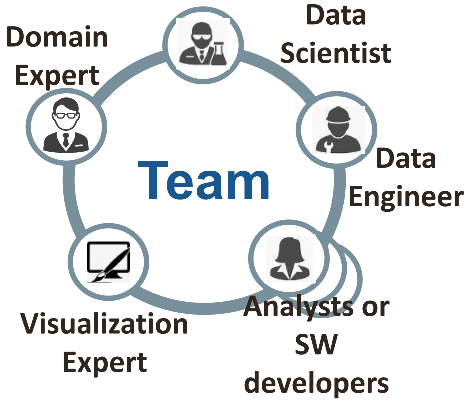



Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

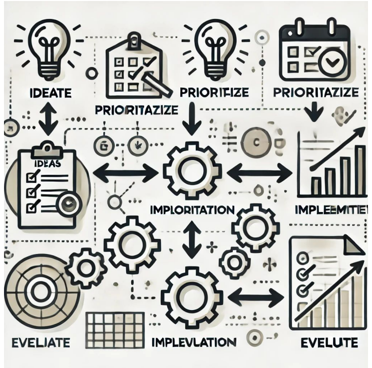
Sculley, et. al. "Hidden Technical Debt in Machine Learning"
Advances in Neural Information Processing Systems 28 (NIPS 2015)

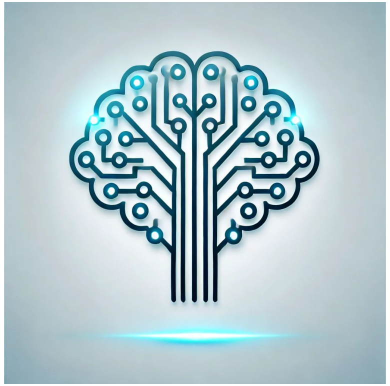
| Requirements | Operationalize Data | Analytic Method | Deployment |
|---|---|---|---|
| Understand the problem | Collect, Move, Store? | Algorithm & method selection | Design output, visualization, automation |
| Identify gaps where AI can assist | Explore, Transform, Harmonize | Hyperparameter and model training | Continuous monitoring for concept drift |
| Decision, automation, interaction? | Label, Learn, Optimize | Assess bias, verify, validate, interpretation | Model Retraining Strategy |
| Domain Expert | Data Engineer | Data Scientist | Visual Expert |



# Key Steps



Ideation Surrounding:
- Business strategy
- Pain points
- Sales differentiators
- Quick wins

Environment & Access:
- Infrastructure
- Data access
- Sandbox/Dev
- Milestones

Collaboration:
- Sprints/demos
- Domain Experts
- Creativity
- Relationships

Deployment:
- Advanced Plan
- Proc. Engr
- Change Mgmt
- RoI

# Artificial Intelligence Uses

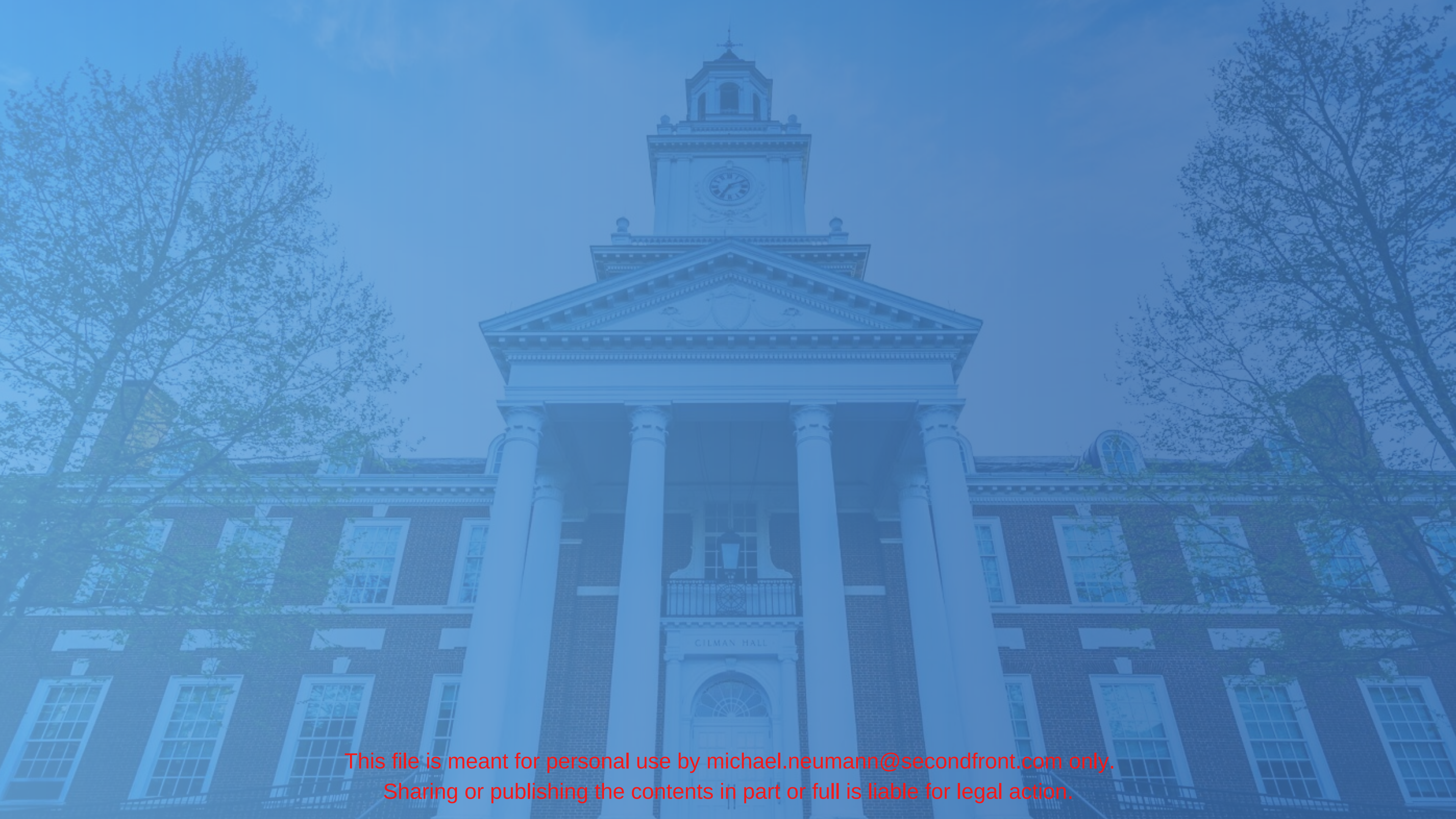
Decision Making

Automation

Personalized Interaction

Responsible AI

**AI must always support business value…not the other way around.**


GILMAN HALL

# Problem Space

## Real-World Applications

# Introduction

**Key Focus Areas:**

- Interconnected nature of industry problems
- Application of machine learning to real-world challenges

**Industries Covered:**

- Healthcare
- Defense
- Banking

# Industry 1: Healthcare

**Key Challenges:**

- Disease diagnosis
- Hospital operations efficiency
- Patient risk prediction

**ML/AI Techniques:**

- Deep Learning: Medical imaging analysis for accurate diagnosis
- RFID Tracking: Automating infusion pump inventory management
- Gradient Boosting Algorithms: Predicting patient readmissions

# Industry 1: Healthcare - Challenges and ML/AI Applications

# Industry 1: Healthcare - Challenges and ML/AI Applications

**Disease Diagnosis:**

- Neural networks for X-rays and MRI analysis
- Reducing physician workload and improving outcomes

# Industry 1: Healthcare - Challenges and ML/AI Applications

**Disease Diagnosis:**

- Neural networks for X-rays and MRI analysis
- Reducing physician workload and improving outcomes

**Hospital Operations:**

- RFID and AI to optimize infusion pump distribution
- Social connections among staff to enhance resource sharing

# Industry 1: Healthcare - Challenges and ML/AI Applications

**Disease Diagnosis:**

- Neural networks for X-rays and MRI analysis
- Reducing physician workload and improving outcomes

**Hospital Operations:**

- RFID and AI to optimize infusion pump distribution
- Social connections among staff to enhance resource sharing

**Patient Risk Prediction:**

- Analyzing structured data to reduce complications and save costs

# Industry 2: Defense

**Key Challenges:**

- Threat detection
- Automating high-risk tasks
- Enhancing soldier performance

**ML/AI Techniques:**

- Computer Vision: Identifying threats from imagery
- Robotics: Autonomous bomb disposal and demining
- Time-Series Models: Real-time monitoring of soldier health

# Industry 2: Defense - Challenges and ML/AI Applications

# Industry 2: Defense - Challenges and ML/AI Applications

**Threat Detection:**

- Using AI to identify precursors to attacks (e.g., illicit supply chains)

# Industry 2: Defense - Challenges and ML/AI Applications

**Threat Detection:**

- Using AI to identify precursors to attacks (e.g., illicit supply chains)

**Automation:**

- Deploying robots for roadside bomb disposal and demining

# Industry 2: Defense - Challenges and ML/AI Applications

**Threat Detection:**

- Using AI to identify precursors to attacks (e.g., illicit supply chains)

**Automation:**

- Deploying robots for roadside bomb disposal and demining

**Soldier Performance:**

- Predicting fatigue and cognitive attention using sensor data

# Industry 3: Banking

**Key Challenges:**

- Fraud detection
- Credit scoring
- Customer retention

**ML/AI Techniques:**

- Anomaly Detection: Real-time fraud detection
- Decision Trees: Automated credit scoring
- Personalized AI Systems: Enhancing customer service

# Industry 3: Banking - Challenges and ML/AI Applications

# Industry 3: Banking - Challenges and ML/AI Applications

**Fraud Detection:**

- Improved ML models reducing false positives and financial losses

# Industry 3: Banking - Challenges and ML/AI Applications

**Fraud Detection:**

- Improved ML models reducing false positives and financial losses

**Credit Scoring:**

- AI-driven equitable solutions for unconventional applicants

# Industry 3: Banking - Challenges and ML/AI Applications

**Fraud Detection:**

- Improved ML models reducing false positives and financial losses

**Credit Scoring:**

- AI-driven equitable solutions for unconventional applicants

**Customer Retention:**

- AI-powered call centers reducing wait times and tailoring services

# Key Takeaways

**Interconnected Nature of Problems:**

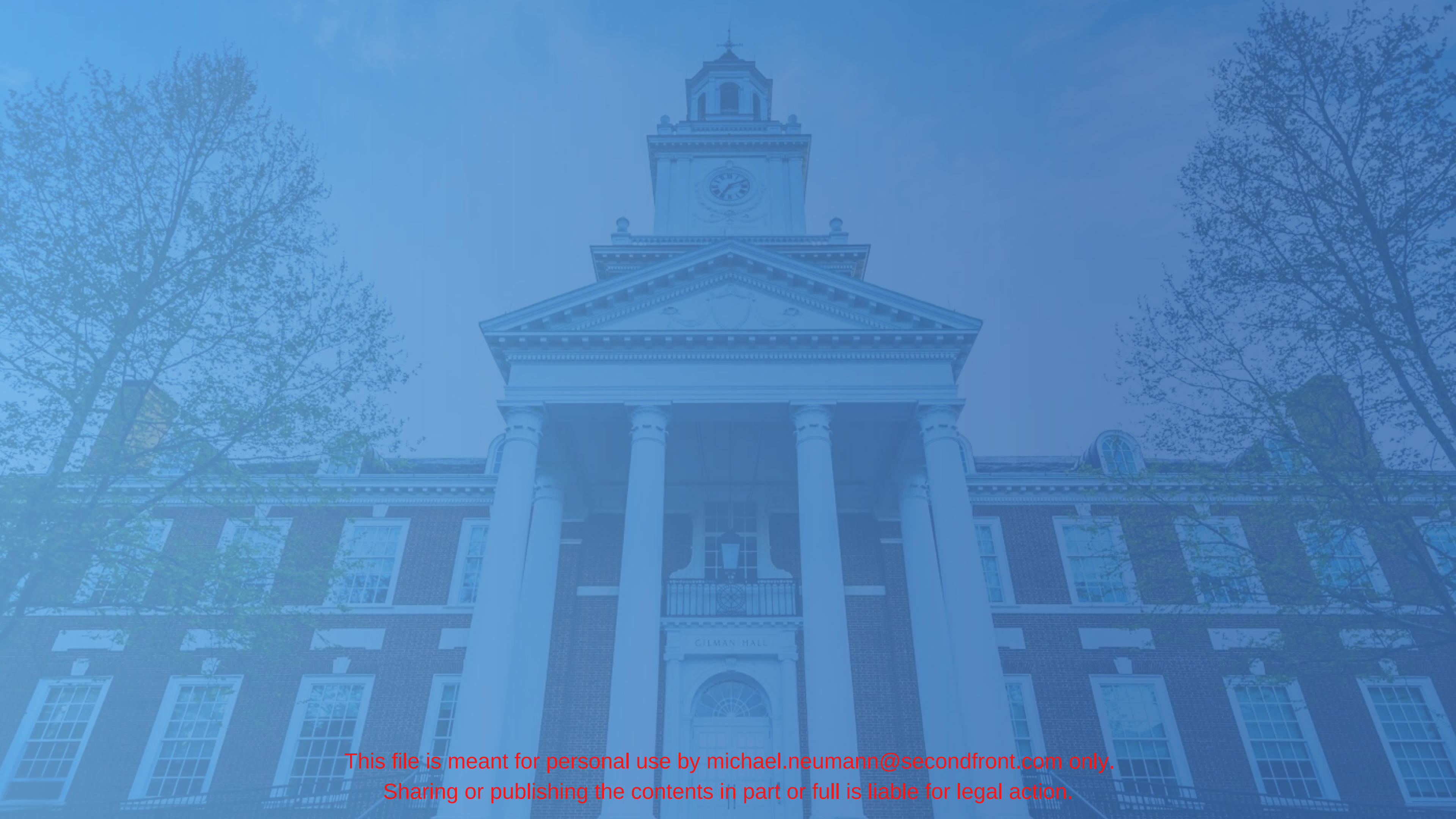
- Solutions in one area often impact others

**Holistic Thinking in ML:**

- Decision support, automation, and personalized interaction are deeply linked

**ML's Power Across Industries:**

- Deep learning for medical imaging
- Reinforcement learning for drones
- Anomaly detection for fraud prevention


GILMAN HALL

# Logistic Regression

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Great Learning POWER AHEAD

# What is Logistic Regression?

**Definition:**

- A classification algorithm predicting probabilities for binary outcomes.

**Examples:**

- Spam or not spam.
- Fraudulent or non-fraudulent transactions.
- Presence or absence of a medical condition.

# How Logistic Regression Works

**Core Concept:**

- Uses a mathematical function — the sigmoid function.

**Process:**

- Input features (e.g., income, browsing history).
- Apply weights to compute a weighted sum.
- Pass through the sigmoid function to generate probabilities (0 to 1).

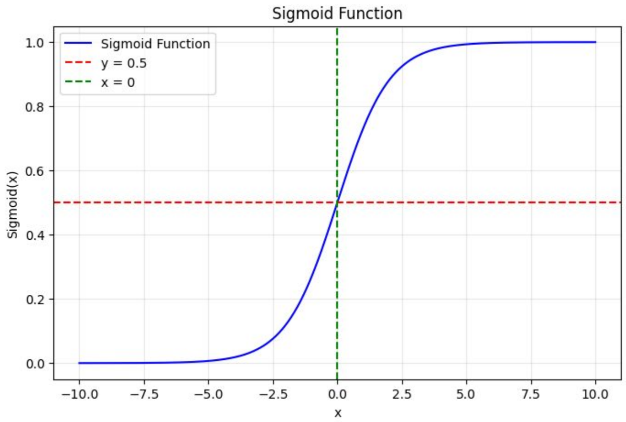
# The Sigmoid Function

**Formula**:

- Converts input (z) into probabilities.
- Weighted sum: z = w1x1 + w2x2 + ... + wn*xn.

**Purpose:**

- Helps classify binary outcomes.



## Decision Boundary

**Default:** Threshold at 0.5.

- Predicts “1” if probability > 0.5.
- Predicts “0” if probability <= 0.5.

**Adjustable**:

- Tune for better precision, recall, or specificity.

**Example:**

- Probability = 0.7; Decision boundary = 0.5 → Predict “Yes”.

# When to Use Logistic Regression

**Ideal Conditions:**

- Linear relationships between input features and target.
- Binary classification problems (e.g., yes/no, fraud/not fraud).
- Clean, well-structured data.

# When to Use Logistic Regression

**Ideal Conditions:**

- Linear relationships between input features and target.
- Binary classification problems (e.g., yes/no, fraud/not fraud).
- Clean, well-structured data.

**Limitations:**

- Non-linear relationships
- High-dimensional data
- Multi-class problems

# Preparing Data for Logistic Regression

**Requirements:**

- Structured tabular data.
- Binary target variables.
- Scaled input features for faster convergence.

# Preparing Data for Logistic Regression

**Requirements:**

- Structured tabular data.
- Binary target variables.
- Scaled input features for faster convergence.

**Challenges:**

- Missing values → Imputation or removal.
- Multicollinearity → Dimensionality reduction.
- Class imbalance → Oversampling or undersampling.

# Optimization Strategies

**Algorithm Adjustments:**

- Learning rate: Controls training speed.
- Regularization: L1 (lasso) or L2 (ridge) to prevent overfitting.

# Optimization Strategies

**Algorithm Adjustments:**

- Learning rate: Controls training speed.
- Regularization: L1 (lasso) or L2 (ridge) to prevent overfitting.

**Data Optimization:**

- Feature engineering: Add interaction terms or polynomial features.
- Noise reduction: Remove irrelevant features.

# Testing and Validation

**Cross-validation:**

- Example: K-fold cross-validation.
- Split data into multiple folds.
- Test model performance on different subsets.

**Outcome**: Provides a robust estimate of model accuracy.

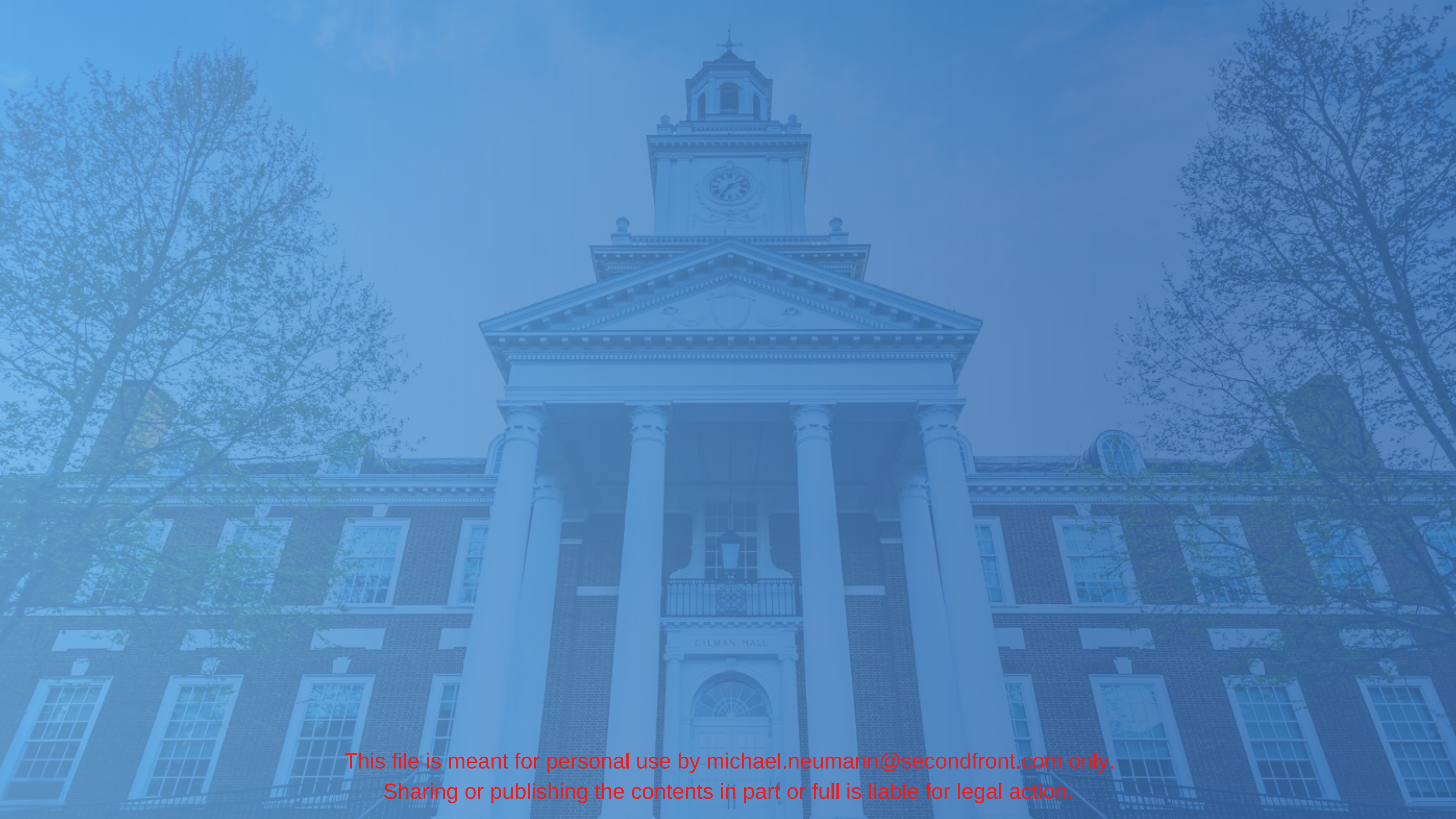
# Summary

### Key Takeaways:

- Logistic regression: Simple, effective for binary classification.
- Best suited for linear, structured, and balanced data.
- Limitations: Non-linearity, multi-class problems, high-dimensional data.

### Next Steps:

- Explore multi-class extensions.
- Compare with other classifiers (e.g., decision trees, SVMs).


GILMAN HALL

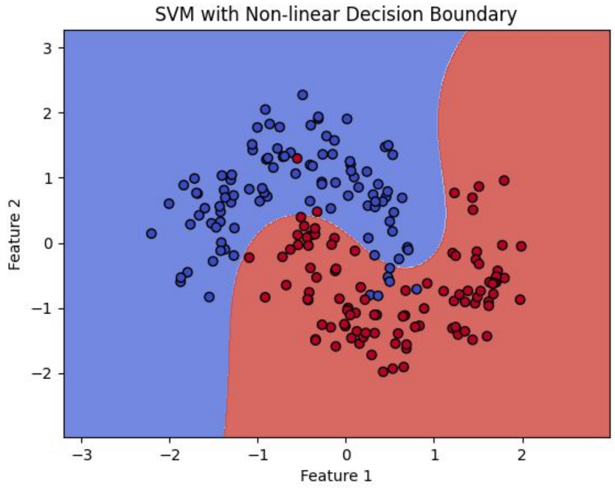
# Support Vector Machines (SVM)

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Great Learning POWER AHEAD

# Introduction: Support Vector Machines

- A versatile ML algorithm for classification and regression.
- Ideal for complex classification problems where data is not linearly separable.
- **Key Strengths:**
  - Handles high-dimensional data.
  - Clear margin separation between classes.
  - Effective for multi-class problems.



# Real-World Applications

**Text & Sentiment Classification**

- Spam detection, product reviews (positive, negative, neutral).

**Image Recognition**

- Optical character recognition, object separation.

**Bioinformatics**

- Protein classification, disease diagnosis (e.g., gene expression data).

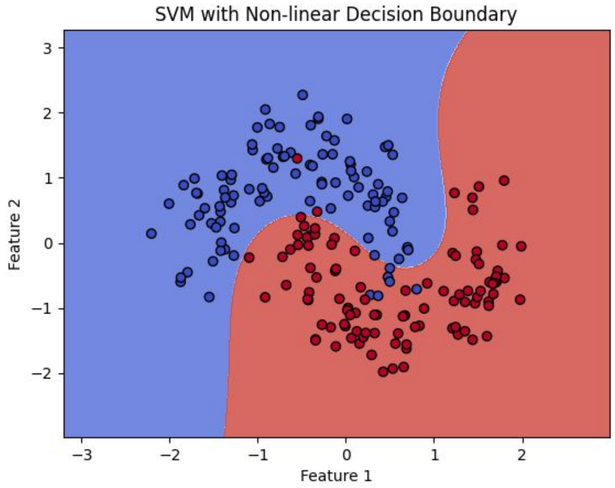
# Core Concepts

**Decision Boundary:**

- 2D: Line | 3D: Plane | Higher dimensions: Hyperplane.

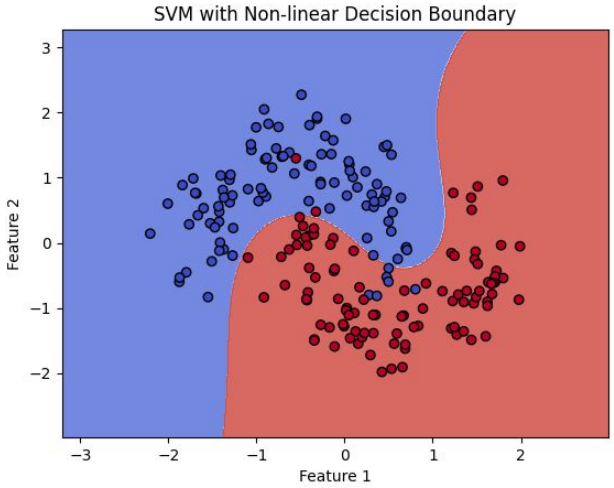
**Margin:**

- Distance between decision boundary and closest data points (support vectors).
- Maximizing the margin reduces misclassification risk and improves generalization.



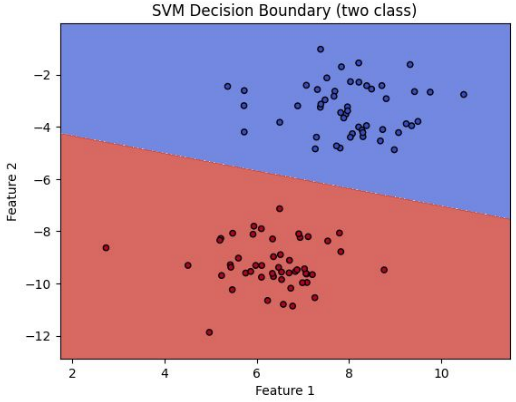
# Linear vs. Nonlinear Classification

- Linear: Directly finds the hyperplane.
- Nonlinear: Uses the Kernel Trick to transform data into higher dimensions for separation.
    - Common Kernels:
        - Linear
        - Polynomial
        - Radial Basis Function (RBF).



# When to Use SVM

- High-dimensional spaces with many features but fewer samples.
- Clear margin of separation between classes (e.g., cancerous vs. non-cancerous cells).
- Small to medium-sized datasets.




## When NOT to Use SVM

- Large datasets (computationally expensive).
- Noisy data (overlapping classes reduce accuracy).
- Multi-class classification (optimized for binary).

# Data Preparation for SVM

- **Structured Data:** Well-defined features required.
- **Standardization & Normalization:** Avoid bias from varying feature magnitudes.
- **Handle Outliers:** Remove or regularize to avoid undue influence.
- **Imbalanced Data:** Use oversampling/undersampling for balanced datasets.

# Hyperparameter Tuning

**Regularization Parameter (C):**

- Balances low error in training vs. larger margin.

**Kernel Selection:** Choose appropriate kernel for the problem.

**Gamma (for RBF Kernel):**

- High gamma: Focus on closer data points.
- Low gamma: Consider distant data points.

**Slack Variables:** Allow flexibility through controlled misclassifications.

# Feature Engineering

- Add relevant features to capture data patterns.
- Eliminate redundant/irrelevant features.
- Balance classes to improve model performance.

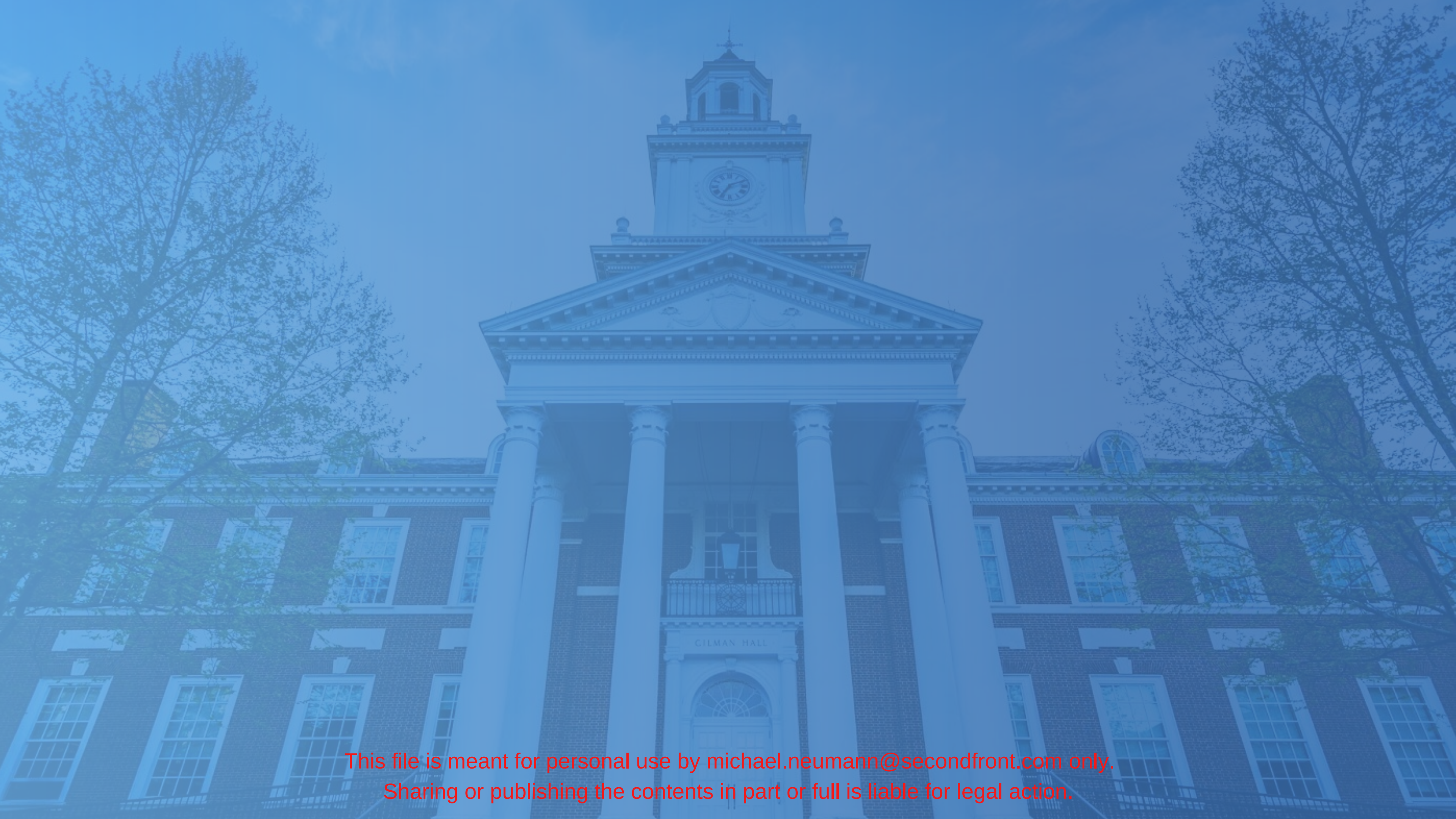
# Summary

**Strengths:**

- High-dimensional data handling.
- Clear class separations.

**Limitations:**

- Requires thoughtful data preparation and hyperparameter tuning.
- Common Use Cases: Text classification, image recognition, bioinformatics.
- Next Steps: Learn tuning strategies for other models like decision trees, random forests, and neural networks.


GILMAN HALL

# Naive Bayes Algorithm

# What is Naive Bayes?

**Definition**:

- A probabilistic algorithm for classification.

**Foundation:**

- Based on Bayes' Theorem.
- Assumes features are conditionally independent.

**Applications**:

- Spam detection
- Sentiment analysis
- Document classification
- Medical diagnosis

# Bayes' Theorem Overview

- **Formula:**

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

- **Components:**

  - $P(C|X)$: Probability of class $C$ given evidence $X$.
  - $P(X|C)$: Likelihood of evidence $X$ given class $C$.
  - $P(C)$: Prior probability of class $C$.
  - $P(X)$: Normalizing constant to make results a probability.

# Steps in Naive Bayes Algorithm

**Step 1: Calculate Prior Probabilities P(C)**

- Determine frequency of each class in training data

**Step 2: Compute Likelihood P(X|C)**

- Calculate probability of feature value X for each class C

**Step 3: Normalize**

- Use P(X) to scale probabilities between 0 and 1.

**Step 4: Classify**

- Compare probabilities to determine the class.

**Example: Spam Detection**

# When to Use Naive Bayes

**Ideal Scenarios:**

- Text classification
- Multiclass problems (e.g., positive, negative, neutral sentiments)
- High-dimensional data
- Small datasets
- When probabilistic output is required

# Limitations of Naive Bayes

**Correlated Features:**

- Independence assumption fails with highly correlated data.

**Continuous Data:**

- Requires Gaussian assumptions or discretization.

**Complex Relationships:**

- Struggles with intricate decision boundaries.

# Preparing Data for Naive Bayes

**Ensure Categorical or Discrete Data:**

- Example: Word counts in text classification.

**Data Cleaning:**

- Handle missing or incorrect values to ensure accurate probability calculations.

**Class Definition:**

- Clear and fewer predefined classes improve performance.

**Avoid Zero Probabilities:**

- Use Laplace smoothing to prevent zero values.

# Improving Performance

**Algorithm Adjustments:**

- Add small constants (e.g., Laplace smoothing).
- Choose appropriate variants: Gaussian, Multinomial, or Bernoulli.

**Data Optimizations:**

- Remove redundant features.
- Balance class distributions using upsampling or downsampling.
- Discretize continuous data.

# Evaluating Naive Bayes

**Use metrics like:**

- Precision, Recall, F1 Score
- ROC Curve and AUC

**Cross-validation:**

- Ensures reliable performance assessment.

# Key Takeaways

**Strengths:**

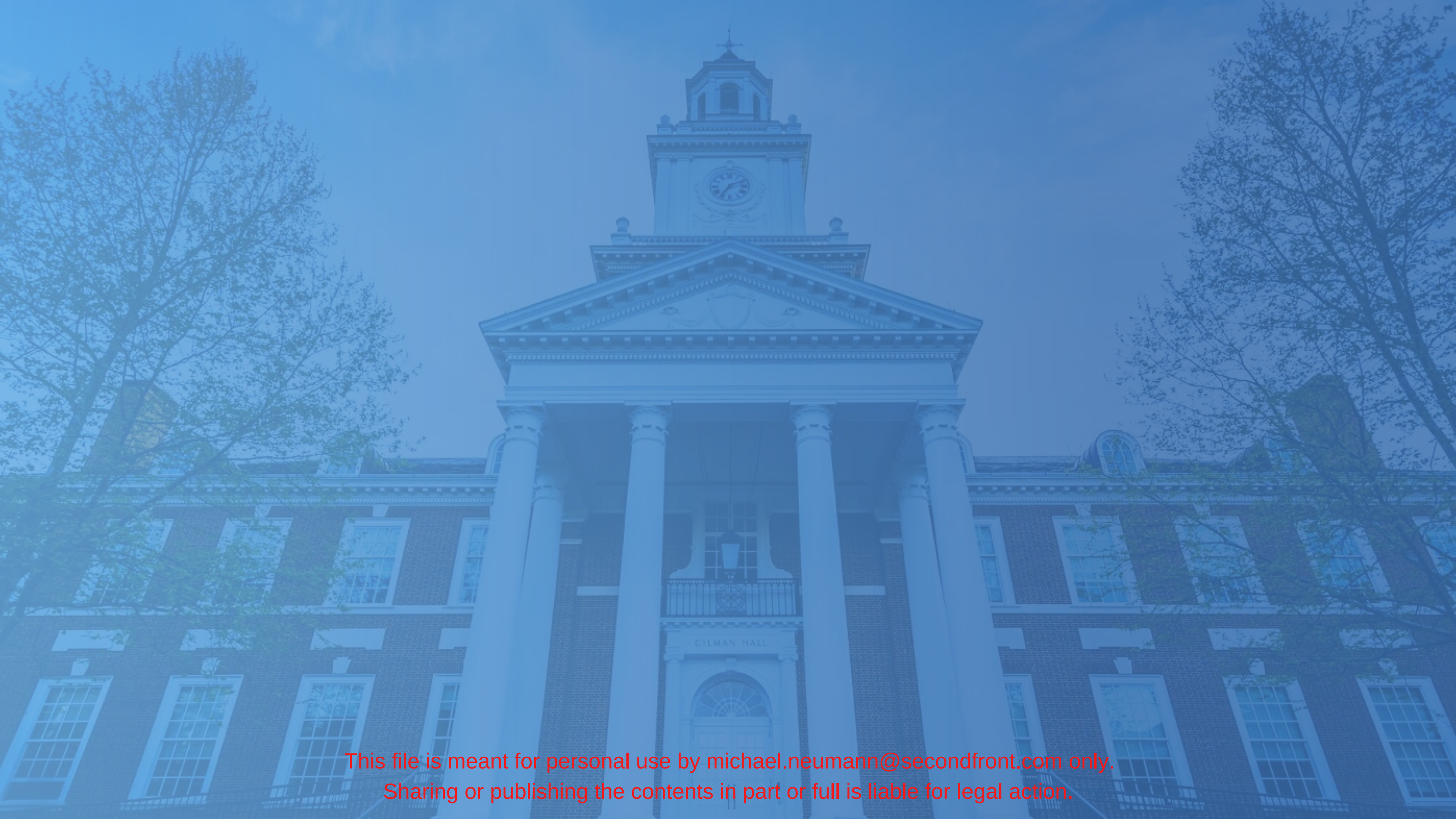
- Simple and effective, especially for text data.
- Performs well with small and high-dimensional datasets.

**Limitations:**

- Independence assumption and continuous data challenges.

**Next Steps:**

- Explore advanced algorithms like Decision Trees and Random Forests.


GILMAN HALL

# Introduction to Decision Trees

# How Decision Trees Work

**Structure**:

- Resembles a flowchart.

**Splitting criteria:**

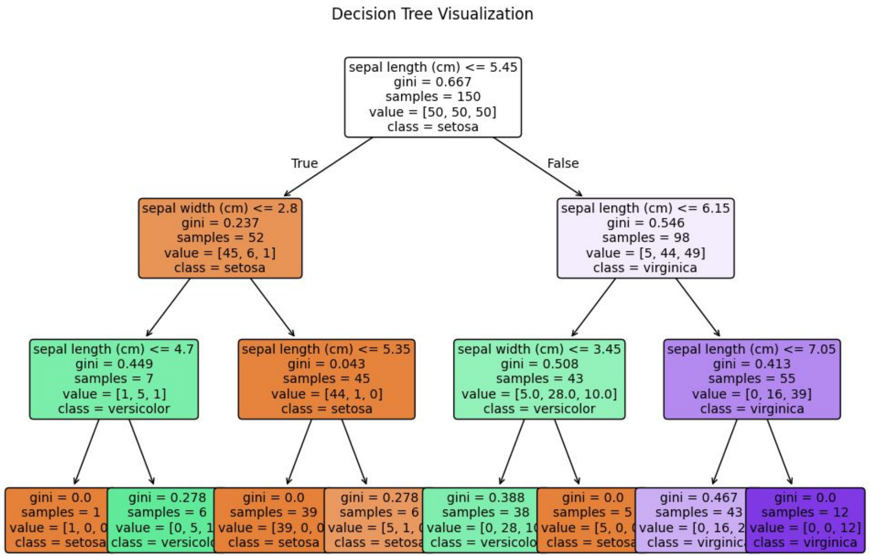
- Metrics: Gini impurity, Entropy/Information gain, Variance reduction.

**Leaf nodes:**

- Final output (class label or regression value).

**Decision paths:**

- Traverse the tree based on feature values.



# Example: Product Purchase Decision

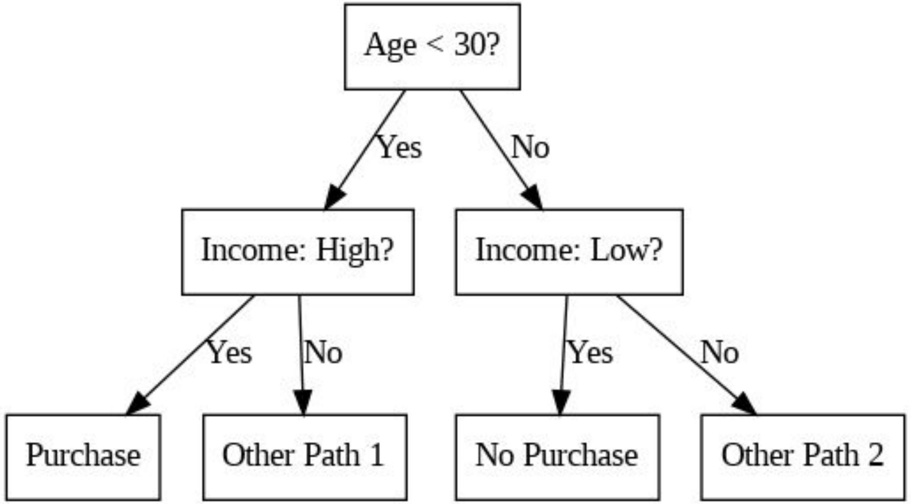
**Features:**

- Age: <30, 30–50, >50
- Income: High, Medium, Low

**Paths:**

- If under 30 & high income → Purchase.
- If over 50 & low income → No purchase.

Decision trees classify based on multiple splits.



# When to Use Decision Trees

**Advantages:**

- Interpretability: Easy to explain and defend.
- Handles structured data well (categorical/numerical).
- Effective for non-linear relationships and small datasets.

**Applications:**

- Medical decisions, financial audits, personalized applications.

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

Great Learning POWER AHEAD

# Limitations of Decision Trees

**Overfitting:**

- Trees that grow too deep memorize training data.

**Instability:**

- Small data changes lead to entirely different trees.

**High dimensionality:**

- Struggles with many features and small datasets.

**Linear relationships:**

- Simpler models like logistic regression may perform better

# Preparing Data for Decision Trees

- **Clean Data**: Handle missing data (impute or remove rows).
- **Feature Scaling:** Not required.
- **Class Imbalance**: Use weighting, upsampling, or downsampling.
- **High Cardinality**: Group similar categories to reduce overfitting.

# Strategies to Improve Performance

**Algorithm Tweaks:**

- Limit tree depth to prevent overfitting.
- Set minimum samples for splits and leaf nodes.
- Impurity thresholds: Avoid negligible splits.
- Pruning: Remove branches with minimal contribution.

# Strategies to Improve Performance

**Algorithm Tweaks:**

- Limit tree depth to prevent overfitting.
- Set minimum samples for splits and leaf nodes.
- Impurity thresholds: Avoid negligible splits.
- Pruning: Remove branches with minimal contribution.

**Data Adjustments:**

- Combine or create new features.
- Address class imbalance (e.g., SMOTE).
- Bin continuous features to improve splits.

# Advanced Techniques

**Ensemble Methods:**

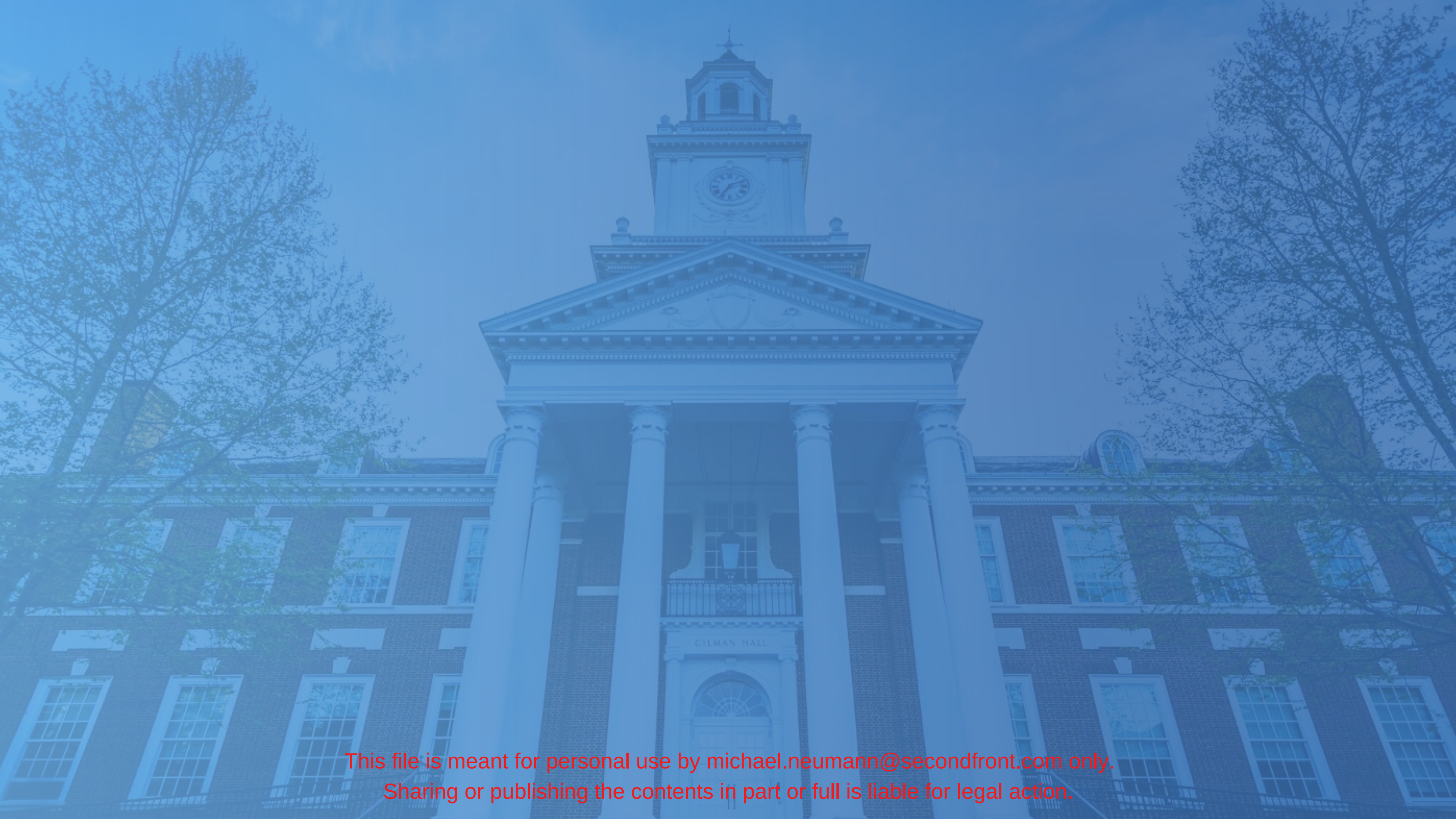
- Random Forest: Combine multiple decision trees.
- Boosting: Correct mistakes (e.g., Gradient Boosting, XGBoost).

**Validation:**

- Use cross-validation to evaluate performance.

# Conclusion

- Decision trees: Powerful and interpretable models.
- Require careful tuning to avoid overfitting and instability.
- Serve as a foundation for ensemble methods like Random Forests and Gradient Boosting.
- Next lecture: Explore Random Forests and Gradient Boosting techniques.


GILMAN HALL

# Random Forest

# What is Random Forest?

**Definition**:

- An ensemble learning method combining multiple decision trees.

**Purpose:**

- Reduce overfitting
- Generalize better to new data
- Handle high-dimensional data efficiently

# Applications of Random Forest

**Customer Churn Prediction**:

- Identifying customers likely to leave.

**Fraud Detection**:

- Enhancing predictions by combining decision trees.

**Medical Diagnosis**:

- Predicting diseases based on symptoms.

**Feature Selection**:

- Identifying the most important variables in large datasets.

# How Random Forest Works

**Bootstrap Aggregation (Bagging):**

- Each tree is trained on a random sample of the data with replacement.
- Introduces diversity, reduces overfitting.

**Random Feature Selection:**

- At each split, considers only a subset of features.
- Reduces correlation among trees.

**Prediction Aggregation:**

- Classification: Majority voting.
- Regression: Average of all tree predictions.

# When to Use Random Forest

**Works best for:**

- High-dimensional data (e.g., genomics, text classification).
- Nonlinear relationships.
- Noise-resistant problems.
- Understanding feature importance.
- Imbalanced datasets (e.g., using class weights or oversampling).

# When NOT to Use Random Forest

- **Large datasets with high latency requirements:** Slower training and predictions.
- **Interpretability challenges:** Harder to explain than a single decision tree.
- **Sparse or featureless data:** Struggles with random noise.
- **Extrapolation:** Poor at predicting beyond training data range.

# Data Preparation for Random Forest

**Structured data:**

- Works best with tabular data.

**Missing values:**

- Can handle natively but imputation may boost performance.

**Addressing class imbalance:**

- Use class weighting or oversampling techniques.

# Optimizing Random Forest Performance

**Key Hyperparameters to Tune:**

- Number of trees (n_estimators): Balance performance and compute time.
- Maximum depth: Prevent overfitting.
- Minimum samples to split: Control tree growth.
- Maximum features: Reduce correlation, diversify trees.
- **Out-of-Bag Error:** Built-in validation mechanism for error estimation.

# Enhancing Random Forest with Advanced Techniques

**Cross-Validation:**

- Evaluate model performance and tune parameters.

**Boosting:**

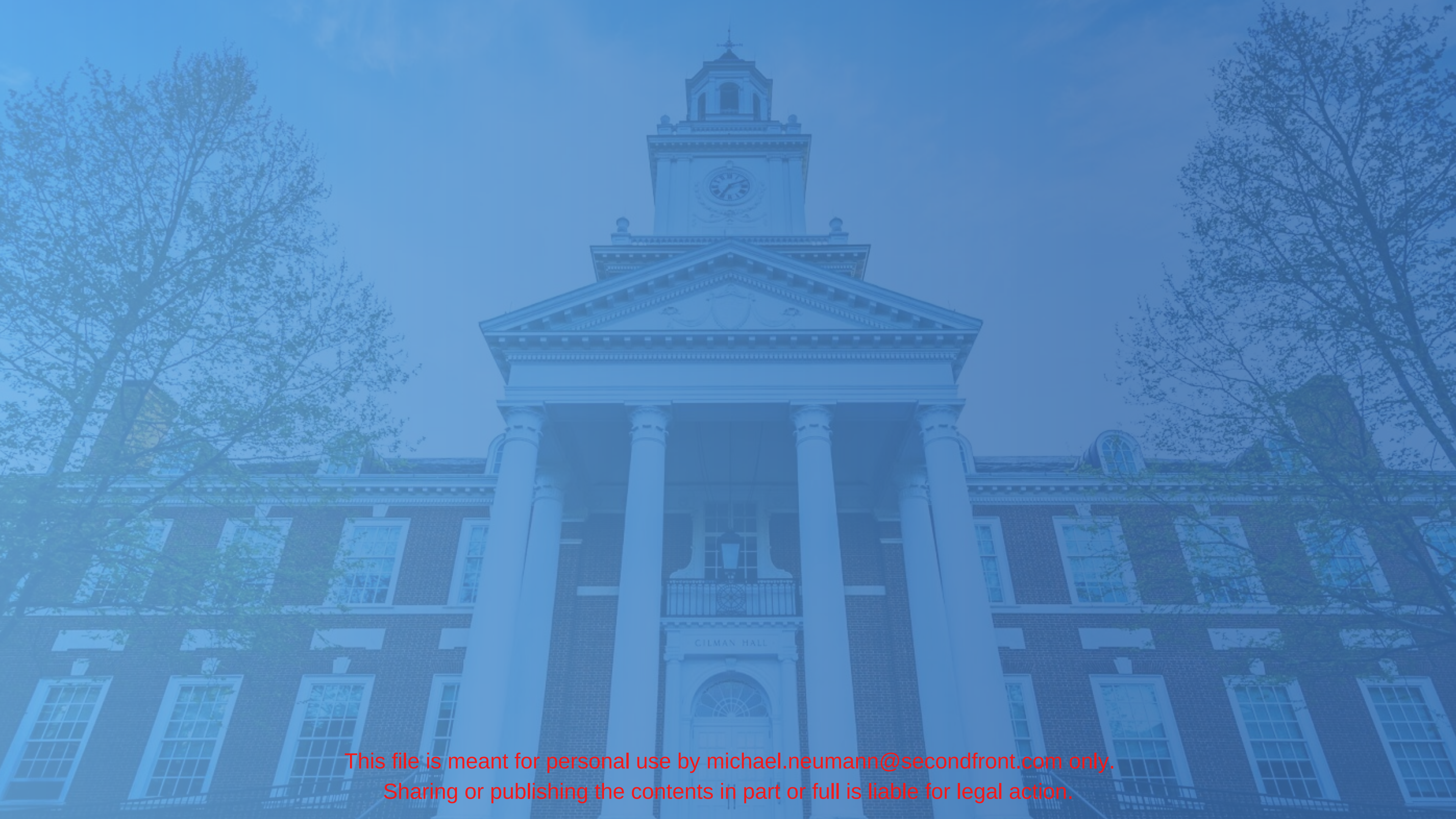
- Combine with methods like XGBoost for enhanced predictions.

**Tree Diversification:**

- Increase randomness by adjusting max features.

# Summary

- Random forest is a versatile tool for classification and regression.
- Balances model complexity with robustness.
- Effective for high-dimensional, imbalanced, or noisy datasets.
- Requires careful tuning and data preparation for optimal results.


GILMAN HALL

# Dimensionality Reduction

# Introduction to Dimensionality Reduction

**Definition:**

- Reduces features/dimensions in a dataset while retaining key information.

**Why It's Important:**

- Handles high-dimensional datasets.
- Reduces computational complexity and overfitting.
- Improves interpretability for humans and models.

# Challenges of High-Dimensional Data

**Curse of Dimensionality:**

- Increased sparsity.
- Harder to identify meaningful patterns.

**Risks:**

- Overfitting irrelevant features.
- Difficulty in visualization beyond 3D.

# Techniques: Principal Component Analysis (PCA)

**What It Does:**

- Identifies linear combinations of features with maximum variance.

**How It Works:**

- Compute covariance matrix.
- Calculate eigenvectors and eigenvalues.
- Select top components.

**Benefits:**

- Retains variability while reducing dimensions.
- Enhances interpretability and speeds up computation.

# PCA Applications

**Facial Recognition:**

- Reduce high-dimensional image data.

**Image Compression:**

- Efficiently encode features for machine learning.

# Techniques: T-SNE (T-Distributed Stochastic Neighbor Embedding)

**What It Does:**

- Visualizes high-dimensional data in 2D/3D.
- Preserves local structure and patterns.

**How It Works:**

- Compute pairwise similarity in high-dimensional space.
- Map points to lower dimensions.
- Optimize to minimize divergence.

**Benefits:**

- Great for visualizing clusters and nonlinear patterns.

# T-SNE Applications

**Word Embeddings:**

- Explore relationships in natural language data.

**Genomics:**

- Identify gene expression patterns.

# When to Use PCA vs. T-SNE

**PCA:**

- Linear transformations.
- High correlation among features.
- Example: Financial data analysis.

**T-SNE:**

- Visualization-focused.
- Complex, nonlinear relationships.
- Example: Image and word embeddings.

# Practical Considerations

**Choosing Dimensions:**

- PCA: Use explained variance ratio.
- T-SNE: Typically set to 2 or 3.

**Scaling/Normalization:**

- PCA: Requires z-score normalization.
- T-SNE: Normalize to a 0-1 range.

**Computational Cost:**

- T-SNE is slower; consider alternatives like UMAP.

# Advantages of Dimensionality Reduction

- Improved model performance: Reduces noise.
- Faster computation.
- Better interpretability: Simplifies data for clearer insights.

# Real-World Applications

**Healthcare:**

- Genomic data analysis for faster insights.

**Finance:**

- Risk modeling and feature selection for loans.

**E-Commerce:**

- Visualize consumer behavior to tailor marketing.

# Summary

**Dimensionality Reduction:**

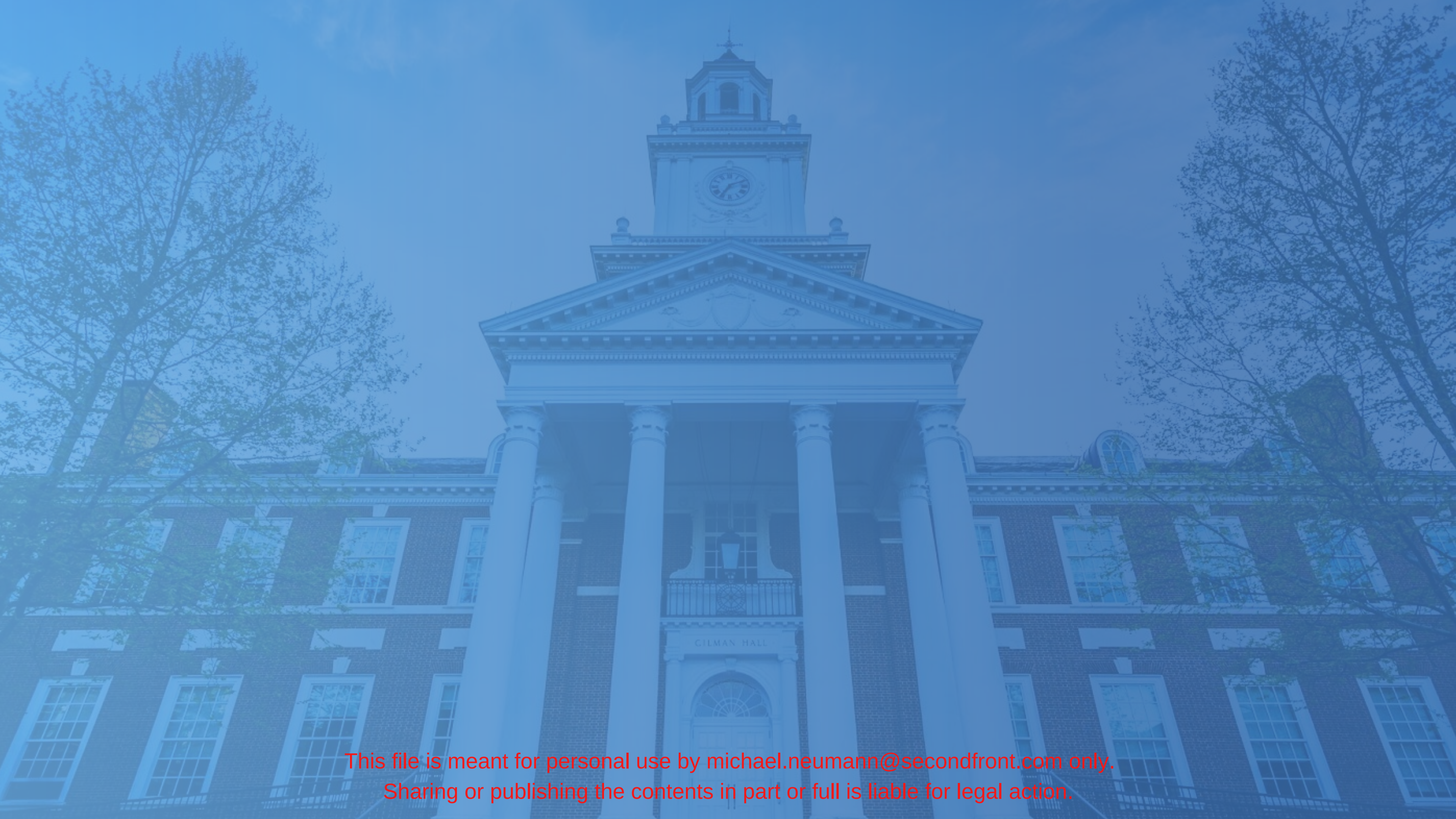
- Simplifies high-dimensional data while retaining meaning.

**Key Techniques:**

- PCA and T-SNE.

**Key Benefits:**

- Enhances model efficiency, reduces overfitting, and improves insights.



# Neural Network

# Neural Network

**Advantage:**

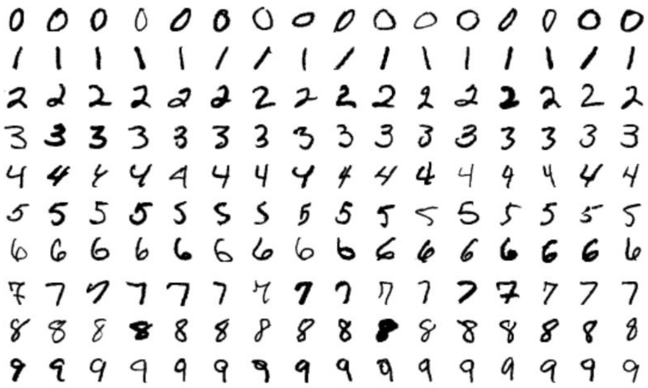
- High Accuracy: Particularly for large datasets and complex problems
- Ability to model complex relationships: Can capture non-linear pattern in data
- Versatile architecture: Various architectures (CNNs, RNNs, etc.) for different tasks

**Disadvantage:**

- Require Large datasets: Performance improves with the amounts of data, which can be a limitation
- Computationally expensive: Training deep networks requires significant computational resources
- Black-box nature: Harder to interpret and understand the decision making process compared to other algorithms

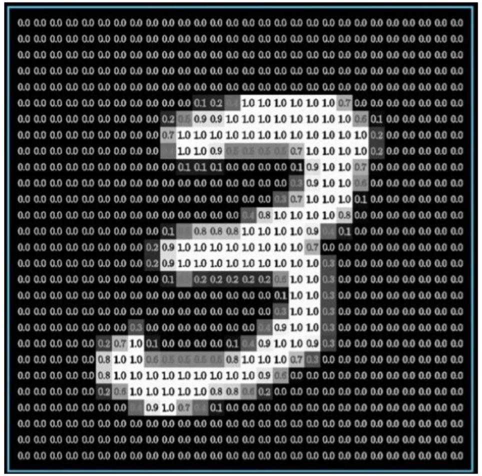
Yann Lecun, Yoshua Bengio, Geoffrey Hinton. Deep learning. Nature, 2015, 521 (7553), pp.436-444. ff10.1038/nature14539ff. ffhal-04206682f

# The Classic Example: MNIST



Modified National Institute of Standards and Technology (MNIST)

24 pixels

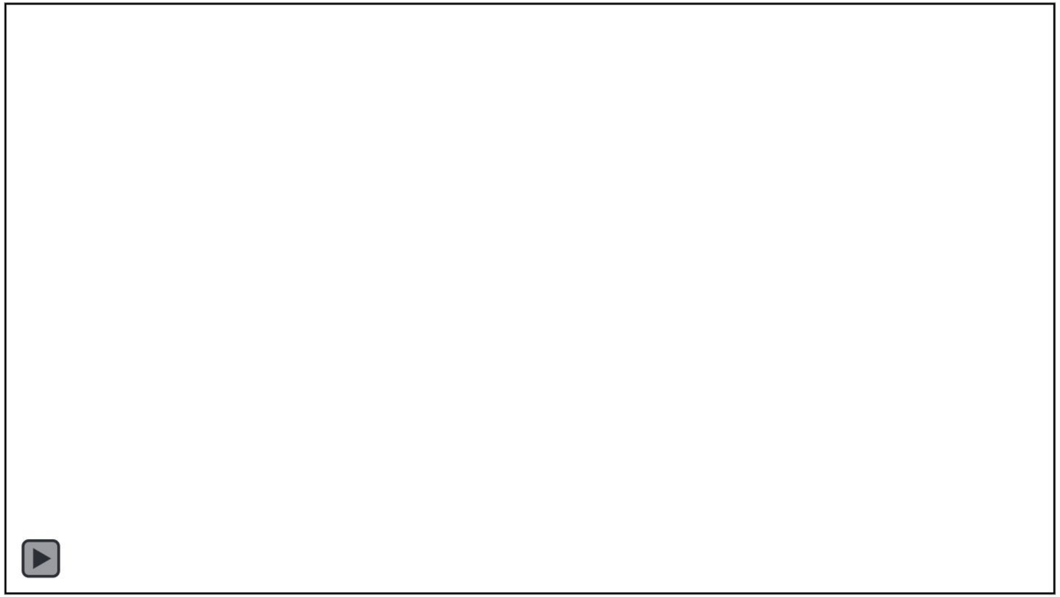


24 pixels

24x24 = 576 pixels

MNIST. (2024, January 27). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=MNIST_database&oldid=1199732782
3blue1brown. (2017, October 5). *But what is a neural network? | Chapter 1, Deep learning* [Video]. YouTube. https://www.youtube.com/watch?v=aircAruvnKk

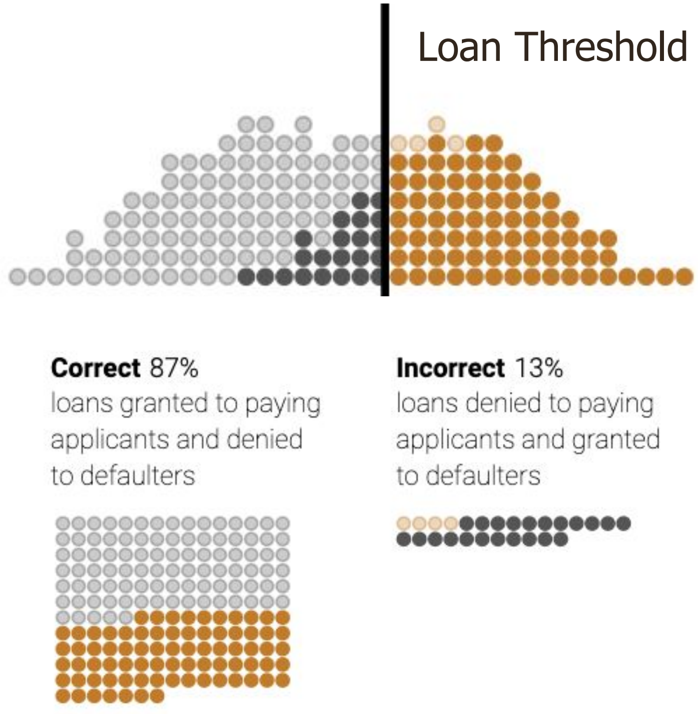
# Supervised Example: Predicting with a Deep Neural Network



Animation from 3Blue1Brown video on Neural Networks. Rebuilt in manim: https:// github.com/3b1b/manim

# AI Performance Measures

# Performance




Define ***Positive*** as successful payment
***Negative*** as default

True Positive = 91
False Positive = 4

True Negative = 95
False Negative = 22

Accuracy = (TP + TN) / (TP + FP + TN +FN)
= (91 + 95) / 212 = 186/212 = 87.7%

Precision is when we say it's a good loan, its good!

Precision = TP / (TP + FP) = 91/(91+4) = 95.8%

Recall is do we grant all good loans

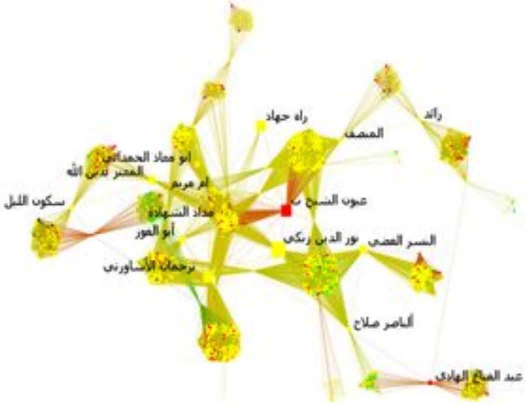
Recall = TP / (TP + FN) = 91/(91+22) = 80.5%

F1 is the "harmonic mean" of precision & recall

F1 = 2(Prec*Recall)/(Prec + Recall) = 87.5%

# Expert Humans Often Disagree

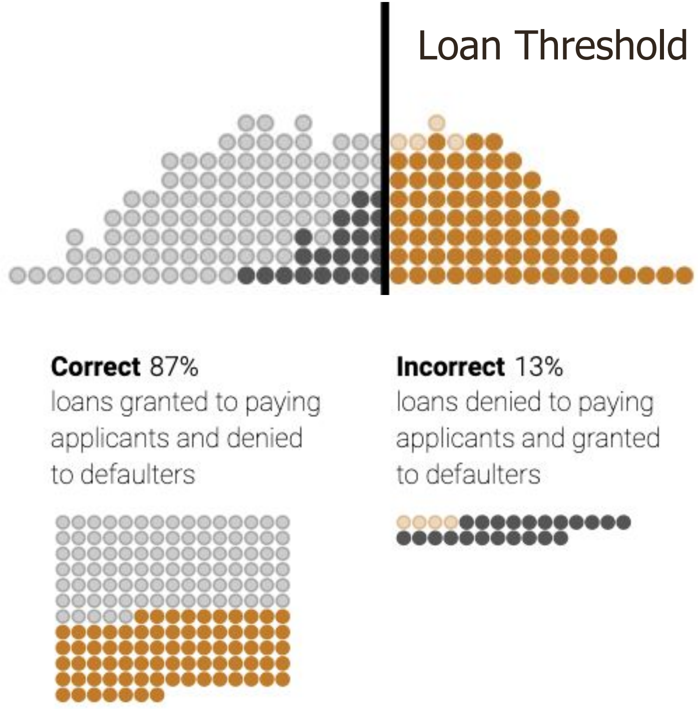
Address resolution
Disagreement = 17%



Online Extremism
Disagreement = 32%

Hypertension Claim
Missing Data = 72%

# Performance




Define ***Positive*** as successful payment
***Negative*** as default

True Positive = 91
False Positive = 4

True Negative = 95
False Negative = 22

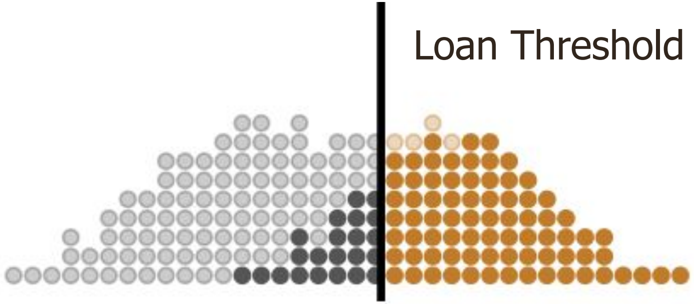
Specificity is do we deny all bad loans

Specificity = TN / (TN + FP) = 95/(95+4) = 96%

| Accuracy | Precision | Recall | F1 | Specificity |
|---|---|---|---|---|
| 88% | 96% | 81% | 88% | 96% |

# Performance

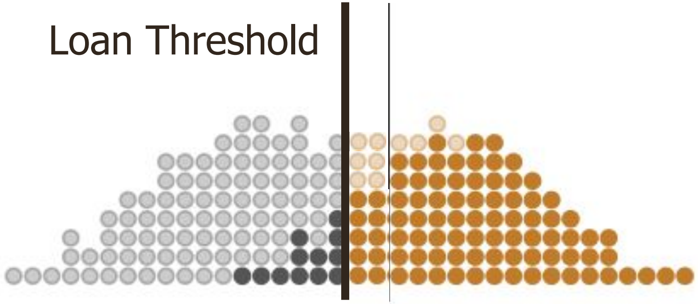
| Accuracy | Precision | Recall | F1 | Specificity |
|---|---|---|---|---|
| 88% | 96% | 81% | 88% | 96% |




True Positive = 101
False Positive = 10

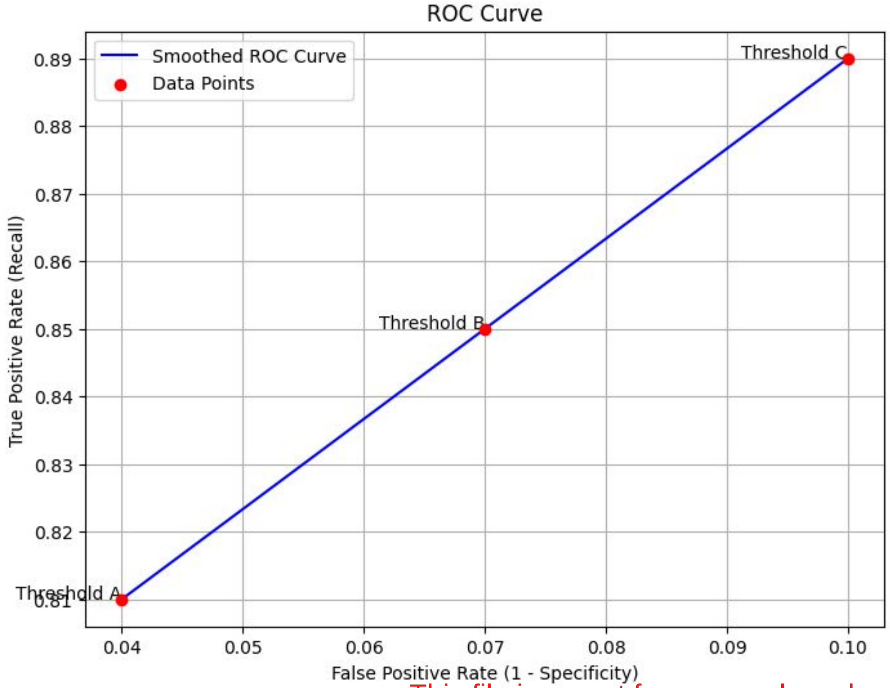
True Negative = 89
False Negative = 12

| Accuracy | Precision | Recall | F1 | Specificity |
|---|---|---|---|---|
| 90% | 91% | 89% | 90% | 90% |



# Performance

## Receiver Operating Characteristic



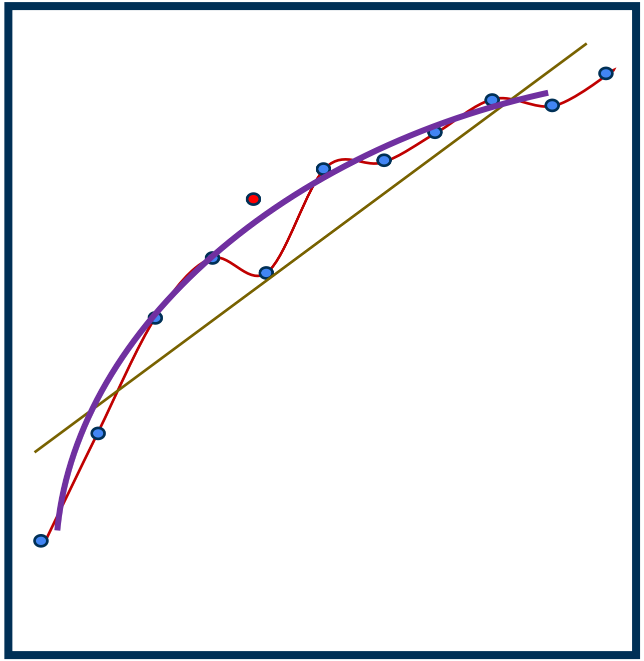

| Accuracy | Precision | Recall | F1 | Specificity |
|---|---|---|---|---|
| 88% | 96% | 81% | 88% | 96% |

True Positive = 101 True Negative = 89
False Positive = 10 False Negative = 12

| Accuracy | Precision | Recall | F1 | Specificity |
|---|---|---|---|---|
| 90% | 91% | 89% | 90% | 90% |

## Area Under the Curve (AUC)

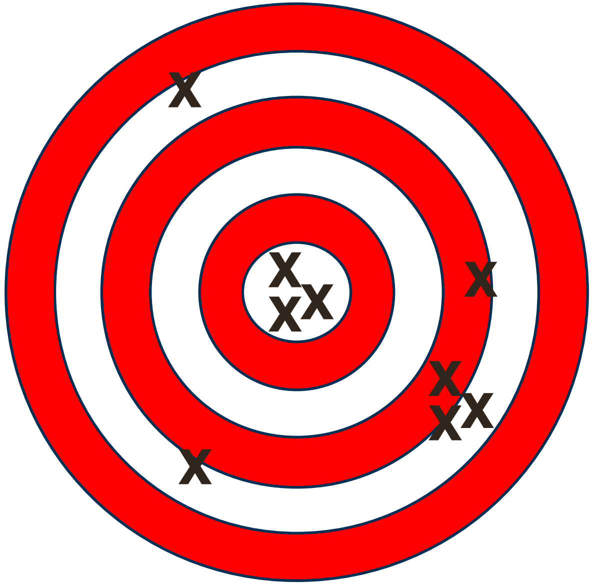
# Overfitting & Underfitting



**Overfitting:** Model learns pattern & noise. It learns the test and doesn't generalize.

**Underfitting:** Model is too simple to capture underlying patterns in data.

# Bias & Variance



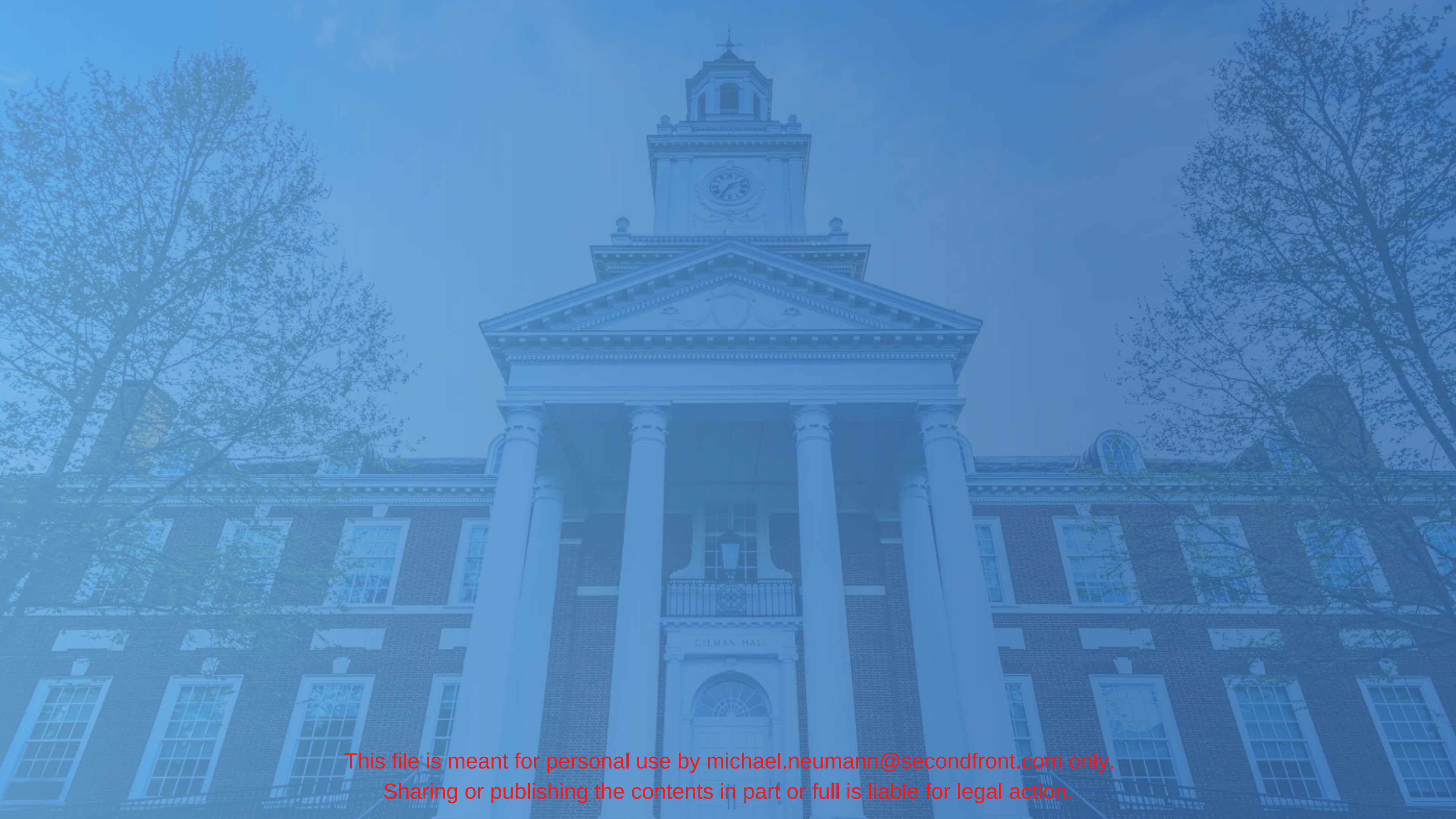
**Bias:** Systematic error approximating a complex model with simple parameters

**Variance:** Model too sensitive to small fluctuations in training data.

**High Bias + Low Variance**: Underfitting.

**Low Bias + High Variance**: Overfitting.

**Low Bias, Low Variance**: Ideal model.


GILMAN HALL