1.

Learning Paradigm	Model Type	Model	• Scenarios to Maximize Effectiveness
Supervised	Discriminative	k-Nearest Neighbors (kNN)	• Instances map to points in n$\Re$. • The average number of attributes is less than 20 per instance. • There is a lot of training data. • The target function is complex but can be approximated by separate local simple approximations
		Linear Regression	• when there is a linear relationship between the predictor and outcome variables
		Logistic Regression	• The dependent variable is discrete (binary outcomes). • A linear decision boundary is appropriate
		Support Vector Machines	• Working with high-dimensional data with very few samples. • The data is linearly separable or can be transformed into a higher-dimensional space where it becomes linearly separable using kernel functions. • Robustness to outliers and noise is desired
	Generative	Naïve Bayes	• Training needs to be easy and fast, only requiring considering each attribute in each class separately. • Testing is straightforward, involving looking up tables or calculating conditional probabilities with normal distributions. • The independence assumption holds or when performance is competitive even with violations of this assumption
		Gaussian Mixture Models (GMMs)	• The data can be represented as a mixture of Gaussian distributions. • Soft clustering is desired, allowing data points to have probabilities of membership in multiple clusters.
Unsupervised	Discriminative	No commonly studied models	• N/A
	Generative	K-Means Clustering	• The data can be partitioned into well-separated clusters, • The number of clusters, k, is known in advance
		Hierarchical Clustering	• A hierarchical representation of the data is desired • The number of clusters is not known in advance
		Kernel K-Means	• The data is not linearly separable and requires non-linear decision boundaries for clustering
		Principal Component Analysis (PCA)	• Reducing the dimensionality of the data while minimizing information loss • Identifying the principal components that capture the most variance in the data
		BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)	• Large Datasets • Minimizes I/O Costs

		Mean-Shift Clustering	• Data points are sampled from an underlying PDF • The goal is to find the densest regions in the feature space

2.

Shallow Neural Network	Objectives	Scenarios Where Most Effective
Single-Neuron Perceptron	Classify linearly separable data.	Useful for simple binary classification tasks where the data is linearly separable. Examples include: Logic gates like AND, OR, and NOT.
Multi-Neuron Perceptron	Extend the single-neuron perceptron to handle multiple inputs and outputs.	Can be used for more complex binary classification tasks with multiple inputs and outputs, as long as the data is linearly separable.
Hamming Network	Determine the class of an input vector by comparing it to a set of prototype vectors.	Useful for pattern recognition tasks where the goal is to find the closest match to a set of known patterns.
Hopfield Network	Store and recall patterns.	Useful for associative memory tasks and pattern completion.
Linear Associator	Learn associations between input and output patterns.	Can be used for tasks such as pattern recognition and image reconstruction.
LMS (Least Mean Squares) Network	Minimize the mean squared error between the network's output and the desired output.	Useful for regression tasks, such as predicting a continuous value. Examples include: Signal processing, like restoring EEG signals
Multilayer Perceptron	Learn non-linear relationships between inputs and outputs.	Useful for a wide range of tasks, including: Classification, such as image recognition and natural language processing. Regression, such as predicting stock prices or housing prices. Function approximation.
Radial Basis Function Network (RBFN)	Approximate functions and perform classification tasks.	Can be applied to various tasks like: Pattern classification, Function approximation, Time series prediction
Extreme Learning Machine (ELM)	Provide fast and efficient learning for single-hidden layer feedforward neural networks.	Offers advantages in scenarios requiring rapid training and good generalization capabilities, particularly in: Classification & Regression.

General Effectiveness of Shallow Neural Networks:

Shallow neural networks are most effective in scenarios where:

- The relationship between inputs and outputs is relatively simple. They may not perform as well as deep neural networks on complex tasks with high-dimensional data.
- The amount of training data is limited. Shallow networks have fewer parameters than deep networks, which makes them less prone to overfitting when trained on smaller datasets.
- Fast training and inference times are required. They are generally faster to train and use than deep networks.

3.

Technique	Application	Strengths	Weaknesses
Feature Selection	Used to reduce the number of features in a dataset by selecting a subset of the most relevant features. (Filtering, Wrapper, Embedded).	• Reduces overfitting risk by reducing model complexity • Reduces dimensionality, leading to faster training and inference times • Improves compute speed • Easier to interpret resulting models	Feature selection is not always necessary
Feature Transformation	Aims to transform the data from a high-dimensional space to a space of fewer dimensions. (Linear (e.g., PCA), Non-linear (e.g., ISOMAP, LLE, t-SNE, UMAP))	• Reduces time complexity due to less computation • Reduces space complexity because of fewer parameters • Saves the cost of observing the feature • Simpler models are more robust on small datasets • More interpretable models, leading to simpler explanations • Data visualization is easier if plotted in 2 or 3 dimensions	Often results in information loss
Principal Component Analysis (PCA)	Seeks a projection that preserves as much information in the data as possible (in a least-squares sense). Finds a new set of dimensions (principal components) that are linear combinations of the original features. The principal components are ordered by the amount of variance they explain in the data. Applications: Face image recognition.	• Relatively simple to implement. • Can be used to reduce the dimensionality of data while preserving most of the variance. • Can be used for data visualization.	• Can be sensitive to outliers. • The principal components may not be interpretable. • Assumes that the data is linearly separable.
Linear Discriminant Analysis (LDA)	Seeks a projection that best separates the data (in a least-squares sense). Finds a new set of dimensions that maximize the between-class scatter and minimize the within-class scatter. Applications: Face recognition with varying expressions and lighting conditions.	• Can be used to reduce the dimensionality of data while preserving class separability. • Can be used for data visualization.	• Assumes that the data is normally distributed. • The number of dimensions that can be reduced is limited by the

			number of classes.
Factor Analysis	Explains the variance among observed variables in terms of a potentially lower number of unobserved variables called factors. Applications: Identification of underlying factors: Clusters variables into homogeneous sets, creates new variables (i.e., factors), and provides insights into categories. Screening of variables: Identifies groupings to select representative variables, useful in regression to address collinearity.	• Reduces the number of variables by identifying underlying factors. • Improves interpretability by grouping correlated variables.	• Subjectivity in factor interpretation. • Assumptions about data normality and linearity may not always hold. • The number of factors to retain can be arbitrary.
Multidimensional Scaling (MDS)	Creates a low-dimensional representation of data based on pairwise dissimilarities between objects. Applications: Visualizing similarities between objects: creating a 2D map to visually confirm findings.	• Can be used to visualize data in a low-dimensional space. • Can handle both metric and non-metric data.	• Can be sensitive to the choice of distance metric. • May not be as accurate as other dimensionality reduction techniques.
Partial Least Squares (PLS)	A supervised dimensionality reduction technique that finds linear combinations of predictor variables that are maximally correlated with the response variable. Applications: Predicting a set of outputs in relation to a reduced order of inputs.	• Can handle data with high dimensionality and multicollinearity. • • Suitable for both regression and classification tasks.	• More complex to interpret than PCA. • Requires careful selection of the number of latent components.

4.

CLAHE (Contrast Limited Adaptive Histogram Equalization)

Strengths:

- Enhanced Contrast: CLAHE improves the overall contrast of an image, making details more visible, especially in low-light or foggy conditions.
- Adaptive Nature: It adjusts the contrast locally, preserving fine details and preventing over-enhancement in some areas.
- Effective for Low-Contrast Images: It's particularly useful for images with poor contrast, such as medical images or those taken in dim lighting.

Weaknesses:

- Potential for Noise Amplification: In some cases, CLAHE can amplify noise, especially in areas with high-frequency details.
- Computationally Intensive: It can be computationally expensive, especially for large images.
- Sensitivity to Parameters: The results can be sensitive to the choice of parameters, such as the clip limit and tile size.

Super Pixel Segmentation

Strengths:

- Reduced Computational Complexity: By grouping similar pixels into superpixels, it significantly reduces the number of elements to process, speeding up subsequent image processing tasks.
- Preserves Object Boundaries: Superpixels tend to respect object boundaries, making them useful for object detection and segmentation tasks.
- Simpler Representations: It provides a more compact representation of the image, which can be beneficial for feature extraction and classification.

Weaknesses:

- Sensitivity to Image Content: The quality of superpixel segmentation can be affected by image content, such as texture and noise.
- Over-Segmentation or Under-Segmentation: The algorithm may sometimes over-segment or under-segment regions, leading to inaccurate results.
- Dependence on Parameter Tuning: The performance of superpixel segmentation can be sensitive to parameters like the number of superpixels and compactness factor.

5.

Overfitting is a phenomenon in machine learning where a model learns the training data too well (in human terms its like when students memorize info instead of learning concept), capturing noise and random fluctuations instead of the underlying patterns. This results in the model performing too well on the training data but poorly on unseen data, leading to poor generalization.

Common strategies to prevent overfitting in classical ML models:

- Feature Selection and Dimensionality Reduction: Reducing the number of features or dimensions in a dataset can simplify the model and prevent it from learning irrelevant details.
  Techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are used to find lower-dimensional representations of the data that preserve the most important information.
- Regularization: This involves adding a penalty term to the model's objective function, discouraging the model from having large weights and prevents the model from becoming too complex and overfitting the training data. Examples include L1, L2, and ElasticNet regularization.
- Early Stopping: In iterative training algorithms like gradient descent, monitoring the model's performance on a validation set and stopping training when performance starts to degrade can help prevent overfitting.
  This involves splitting the data into training, validation, and test sets.

- Pruning: In decision tree models, pruning involves removing branches that do not significantly improve the model's predictive power. This simplifies the tree and prevents it from memorizing the training data.
- Ensemble Methods: Techniques like bagging (bootstrap aggregating) and random forests combine multiple models to reduce variance and improve generalization.

Cross-validation prevents overfitting by providing a more robust estimate of a model's performance on unseen data. It involves partitioning the data into multiple folds and using different folds for training and evaluation. This helps assess the model's generalization ability and choose the best model or hyperparameters.

Types of Cross-Validation Techniques:

- k-Fold Cross-Validation: The data is divided into k folds, and the model is trained and evaluated k times, each time using a different fold for evaluation and the remaining k-1 folds for training.
- Leave-One-Out Cross-Validation (LOOCV): A special case of k-fold cross-validation where k equals the number of data points. The model is trained on all data points except one and evaluated on the left-out point. This process is repeated for each data point.
- Stratified k-Fold Cross-Validation: Ensures that each fold has a similar distribution of class labels, important for imbalanced datasets.
- Time Series Cross-Validation: Used for time-dependent data, where the training data is always from an earlier period than the validation data.

By using cross-validation, we can estimate the model's performance on unseen data more accurately, tune hyperparameters & compare different models,

6.

Classical ML

- Strengths:
  - Simpler models are more robust on small datasets.
  - Easier to interpret
  - Easier to understand
- Weaknesses:
  - No explicit "model" of the concept being learned
  - struggle to learn complex patterns in data often requiring careful feature engineering
  - Requires keeping all the data
  - “Curse of Dimensionality”

Deep Learning

- Strengths:
  - Can learn tasks that are difficult to formalize
  - Excel at image recognition, speech recognition
  - learn complex patterns in data automatically
- Weaknesses:
  - Requires many hours to train
  - The sources do not explicitly describe the weaknesses of deep learning models.
  - Computationally expensive
  - Requires large amounts of data

7.

Common Sources of Error in Classical Machine Learning Models

- **Overfitting:** Model is too complex, memorizes training data, performs poorly on new data.
  - **Solution:** Simpler models, regularization techniques (L1/L2), early stopping, cross-validation.
- **Underfitting:** Model is too simple, fails to capture underlying patterns.
  - **Solution:** More complex models, feature engineering, different algorithms.
- **Bias:** Model systematically favors certain outcomes.
  - **Solution:** Balanced datasets, careful feature selection, algorithm choice.
- **Variance:** Model is sensitive to small changes in training data.
  - **Solution:** Regularization, more data, ensemble methods.
- **Data Quality Issues:** Noise, missing values, outliers.
  - **Solution:** Data cleaning, imputation, normalization, outlier detection.
- **Feature Engineering:** Poorly chosen or engineered features.
  - **Solution:** Domain knowledge, feature selection, dimensionality reduction.
- **Model Selection:** Choosing an inappropriate algorithm.
  - **Solution:** Experimentation, cross-validation, hyperparameter tuning.

8.

You can narrow down the ideal classical machine learning algorithm (from the ones discussed in class by considering factors such as what type of problem you're solving (what kind of solution you require), the characteristics of the data, and expectations from the performance.

Problem Type:
- **Classification**: Predicting a categorical label (e.g., spam or not spam). Algorithms: kNN, Naive Bayes, Logistic Regression, SVMs, Decision Trees, Random Forests.
- **Regression**: Predicting a continuous value (e.g., housing prices). Algorithms: Linear Regression, Regression Trees, Random Forests.
- **Clustering**: Grouping similar data points together. Algorithms: K-means, GMMs, Hierarchical Clustering.
- **Dimensionality Reduction**: Reducing the number of features while preserving important information. Algorithms: PCA, LDA.

Data Characteristics:
- **Data Size and Dimensionality**: For large datasets with high dimensionality, algorithms like SVMs and random forests may be more suitable. For small datasets, kNN or Naive Bayes might be sufficient.
- **Data Type**: Consider whether your features are numerical, categorical, or a mix of both. Some algorithms work better with specific data types (e.g., kNN for numerical data, Naive Bayes for categorical data).
- **Linearity of Relationships**: If you suspect linear relationships between features and the target variable, linear regression or logistic regression may be good starting points. For more complex relationships, consider non-linear models like SVMs with kernels or decision trees.

Performance Expectations: Consider the desired level of accuracy and the trade-off between bias and variance.

9.

1. **Data Preparation:**
   a. **Cleaning:** Handle missing values, outliers, inconsistencies.
   b. **Feature Engineering:** Create new features, transform existing ones.
   c. **Encoding:** Convert categorical variables to numerical format.
   d. **Scaling:** Normalize numerical features.
2. **Model Selection:**
   a. **Regression:** For continuous target variables.

   b. **Classification:** For discrete target variables.
   c. **Clustering:** To group similar data points.
3. **Model Training:**
   a. **Split data:** Divide into training and testing sets.
   b. **Train model:** Use algorithms like linear regression, logistic regression, decision trees, random forests, etc.
4. **Model Evaluation:**
   a. **Metrics:** Choose appropriate metrics (accuracy, precision, recall, F1-score, RMSE, MAE).
   b. **Cross-validation:** Assess model performance on different data subsets.
5. **Hyperparameter Tuning:**
   a. **Optimize:** Adjust model parameters for best performance.
6. **Model Deployment:**
   a. **Integrate:** Deploy the model into a production environment.
   b. **Monitor:** Continuously monitor performance and retrain as needed.

10.

Bias and Fairness: Models can perpetuate societal biases if trained on biased data.
Privacy: Models may expose sensitive information if not properly secured.
Transparency: Lack of interpretability can hinder trust and accountability.
Misuse: Models can be used for malicious purposes, such as discrimination or surveillance.
Job Displacement: Automation can lead to job losses and economic inequality.

11.

Calculating Metrics:

1. **Accuracy:**
   `Accuracy = (TP_A + TP_B + TP_C + TP_D) / Total Samples`
2. **Precision:**
   For each class X: `Precision_X = TP_X / (TP_X + FP_X_A + FP_X_B + FP_X_C)`
3. **Recall:**
   For each class X: `Recall_X = TP_X / (TP_X + FP_A_X + FP_B_X + FP_C_X)`
4. **F1-Score:**
   For each class X: `F1_Score_X = 2 * (Precision_X * Recall_X) / (Precision_X + Recall_X)`

Total Samples: 25 + 3 + 0 + 2 + 3 + 53 + 2 + 3 + 2 + 1 + 24 + 2 + 1 + 58 + 2 + 13 = 200

Elephant:

- **Precision:** 25 / (25 + 3 + 0 + 2) = 0.833
- **Recall:** 25 / (25 + 3 + 2 + 1) = 0.806
- **F1-Score:** 2 * (0.833 * 0.806) / (0.833 + 0.806) = 0.819

Monkey:

- **Precision:** 53 / (3 + 53 + 2 + 3) = 0.828

- **Recall:** 53 / (3 + 53 + 1 + 58) = 0.465
- **F1-Score:** 2 * (0.828 * 0.465) / (0.828 + 0.465) = 0.595

Fish:

- **Precision:** 24 / (2 + 1 + 24 + 2) = 0.800
- **Recall:** 24 / (2 + 1 + 24 + 1) = 0.828
- **F1-Score:** 2 * (0.800 * 0.828) / (0.800 + 0.828) = 0.814

Lion:

- **Precision:** 58 / (1 + 58 + 2 + 13) = 0.784
- **Recall:** 58 / (1 + 58 + 1 + 13) = 0.793
- **F1-Score:** 2 * (0.784 * 0.793) / (0.784 + 0.793) = 0.788

Overall Accuracy: (25 + 53 + 24 + 58) / 200 = 0.775

12.

13.