Sessional-II Pattern:
There are a Total of 5 Questions in the exam, each covering the course content uniquely.
Some of the ways these questions cover the topics are :

1. Conceptual Understanding & EDA (10 Marks)
2. Model Evaluation & Theory (20 Marks)
3. Evaluation Metrics and their uses in different scenarios (20 Marks)
4. Practical Insights (10 Marks)
5. Computational working and analytical understanding of models (20 Marks)

Good Luck and Happy Learning

1. Conceptual Understanding & EDA (10 Marks)

Question: Explain the purpose of Exploratory Data Analysis (EDA) and demonstrate it on a dataset. **Python Code:**

python
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Sample dataset
data = {'Age': [25, 35, 45, 55, 65],
        'Salary': [40000, 50000, 60000, 70000, 80000],
        'Purchased': [0, 1, 1, 0, 1]}
df = pd.DataFrame(data)

# Summary statistics
print(df.describe())

# Correlation matrix
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# EDA visualization
sns.scatterplot(data=df, x='Age', y='Salary', hue='Purchased')
plt.title('Age vs Salary with Purchase Decision')
plt.show()
```

Key Takeaway: This question tests the student's ability to perform basic EDA, derive correlations, and visually explore patterns in data.

2. Model Evaluation & Theory (20 Marks)

Question: Explain the concepts of overfitting and underfitting. Train a decision tree and discuss its evaluation. **Python Code:**

python
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Data preparation
X = df[['Age', 'Salary']]
y = df['Purchased']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Training a Decision Tree
model = DecisionTreeClassifier(max_depth=2)
model.fit(X_train, y_train)

# Predictions and evaluation
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))

# Theory Explanation:
```

```
# Overfitting: When the model performs well on training data but poorly on
test data.
# Underfitting: When the model performs poorly on both training and test
data.
```

Key Takeaway: This tests the conceptual understanding of model generalization and hands-on evaluation.

3. Evaluation Metrics and their uses in different scenarios (20 Marks)

Question: Differentiate between Precision, Recall, and F1-Score. Demonstrate their calculation. **Python Code:**

python
```
from sklearn.metrics import precision_score, recall_score, f1_score,
confusion_matrix

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

# Precision, Recall, and F1-Score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-Score: {f1}")
```

Key Takeaway: Different scenarios require different metrics (e.g., Precision for fraud detection, Recall for medical diagnostics).

4. Practical Insights (10 Marks)

Question: Provide practical insights on the trade-off between simplicity and complexity in model selection. **Insight:** A simpler model, like Linear Regression, is interpretable but might fail for non-linear data. Complex models, like Neural Networks, can handle intricate patterns but require more data, computational resources, and can overfit. **Python Code (Demonstrating Simplicity):**

python
```
from sklearn.linear_model import LinearRegression
import numpy as np

# Example data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
y = np.array([2.2, 2.8, 3.6, 4.5, 5.1])

# Train Linear Regression
model = LinearRegression()
model.fit(X, y)

# Predictions
predictions = model.predict(X)
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

5. Computational Working & Analytical Understanding (20 Marks)

Question: Explain Principal Component Analysis (PCA) and demonstrate with an example.
Python Code:

python
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Sample data
X = df[['Age', 'Salary']]

# Standardization
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Applying PCA
pca = PCA(n_components=1)
X_pca = pca.fit_transform(X_scaled)

print("PCA Explained Variance Ratio:", pca.explained_variance_ratio_)
print("Reduced Data:\n", X_pca)
```

Numerical Example Explanation: PCA reduces dimensions by capturing the direction of maximum variance. Here, data from two features (Age, Salary) is compressed into one principal component.

Lecture 1, which emphasizes introductory concepts of machine learning, its importance, and applications. Here's how the exam questions can map to the slide's key topics:

1. Conceptual Understanding & EDA (10 Marks)

Question: Define machine learning and explain why learning from data is important. Provide examples from the slides. **Answer:** Machine learning involves programming computers to optimize a performance criterion using example data or past experience (slide 3). It is used when:

- Human expertise doesn't exist (e.g., navigating Mars).
- Solutions need adaptation to cases (e.g., user biometrics).

Example from slides:

- Retail: Customer behavior prediction.
- Bioinformatics: Identifying motifs in sequences.

2. Model Evaluation & Theory (20 Marks)

Question: Describe supervised learning and its applications. **Answer:** Supervised learning uses labeled data to train a model. It involves:

1. Predicting outcomes (e.g., stock prices).
2. Knowledge extraction (e.g., decision support in healthcare).

Example Application from slides:

- Medical diagnosis: Predicting disease progression.
- Finance: Credit scoring and fraud detection (slide 14).

3. Evaluation Metrics and their uses in different scenarios (20 Marks)

Question: Discuss the differences between classification and regression. Provide examples. **Answer:**

- **Classification:** Assigns input into categories. Example: Distinguishing high-risk and low-risk customers based on income and savings (slide 10).
- **Regression:** Predicts continuous outcomes. Example: Predicting car prices based on attributes like mileage and age (slide 13).

4. Practical Insights (10 Marks)

Question: Highlight real-world applications of machine learning in daily life. **Answer:** Machine learning impacts various domains:

1. Web Mining: Search engines (slide 7).
2. Retail: Market basket analysis (slide 9).
3. Telecommunications: Spam filtering and intrusion detection.

5. Computational Working & Analytical Understanding (20 Marks)

Question: Explain the process of building a machine learning system. Use examples from the slides. **Answer:** A machine learning system consists of:

1. Problem Identification: Example - Predicting stock prices.
2. Data Collection & Feature Extraction: Example - Customer transactions for retail behavior.
3. Model Training: Example - Training a classifier for spam detection.
4. Evaluation and Deployment: Example - Deploying spam filters (slide 21).

Certainly! I'll base the exam questions and answers on the **Decision Trees** slide content. Here's how the five questions can be crafted:

1. Conceptual Understanding & EDA (10 Marks)

Question: Explain the classification process and its two main steps with an example.
Answer: The classification process includes:

1. **Model Construction:** Using training data to create a classification model represented as rules, decision trees, or mathematical formulae.
2. **Model Usage:** Applying the model to classify unknown test data while evaluating its accuracy (Slides 2-3).

Example: Training data indicates:

- If `rank = professor` OR `years > 6`, then `tenured = yes`.

For testing, given unseen data like (Jeff, Professor, 4), the model predicts whether `tenured = yes` (Slide 4).

2. Model Evaluation & Theory (20 Marks)

Question: What are decision trees and how are they structured? **Answer:** Decision trees are supervised learning models that represent concepts as trees:

1. Each node corresponds to an attribute.
2. Each branch corresponds to an attribute value.
3. Each leaf node assigns a classification (Slide 10).

Key Feature: Decision trees represent disjunctions of conjunctions of constraints, making them interpretable and easy to use (Slide 13).

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: Explain entropy and information gain. Why are they used in decision trees?
Answer:

- **Entropy:** Measures impurity in data; lower entropy implies easier predictability. $$ H(X) = -\sum p_i \log_2(p_i) $$
- **Information Gain:** Measures the reduction in entropy after a split; helps select the best attribute at each node. $$ \text{Gain}(X) = H(\text{Parent}) - \text{Weighted Sum of H(Children)} $$ (Slides 21-26).

Example Calculation (Slide 40):

- Initial Entropy of dataset with Good (3) and Bad (4) grades: $$ H = -(3/7)\log_2(3/7) - (4/7)\log_2(4/7) = 0.985. $$
- Information Gain for an attribute, say `Hardworking`, is calculated stepwise.

4. Practical Insights (10 Marks)

Question: Highlight the interpretability advantage of decision trees. **Answer:** Decision trees are **comprehensible classifiers** (Slide 6):

- Their structure is intuitive; paths from root to leaf directly translate into rules.
- Example rule: **If** Outlook = Sunny **and** Humidity = High, **then** PlayTennis = No (Slide 36).

This makes them suitable for scenarios requiring transparency, such as healthcare decision support.

5. Computational Working & Analytical Understanding (20 Marks)

Question: Describe the ID3 algorithm for building decision trees. Provide an example.
Answer: The ID3 algorithm uses a **greedy search** (Slide 16):

1. Select the best attribute for the root based on highest information gain.
2. Recursively apply the process for child nodes using associated data.
3. Stop when all data is classified or attributes are exhausted.

Example (Slide 47): For attributes `Hardworking`, `Intelligent`, `Unlucky`:

- Calculate entropy and information gain for each.
- If gains are similar, select arbitrarily (e.g., `Hardworking`).

The resulting tree divides the dataset into meaningful classes (Slide 53).

Got it, Syed! I'll base the exam questions and answers on the **Performance Measures** slide content. Here's how the five questions can be tailored:

1. Conceptual Understanding & EDA (10 Marks)

Question: Define accuracy and error rate in the context of classification. Why are they important in evaluating a model? **Answer:**

- **Accuracy:** Measures the ability of a classification method to correctly determine the class of a randomly selected instance. Formula: $$\text{Accuracy} = \frac{\text{C}}{\text{T}}$$, where T = Total instances, C = Correctly classified instances (Slide 1).
- **Error Rate:** The proportion of instances misclassified. Formula: $$\text{Error Rate} = \frac{\text{T - C}}{\text{T}}$$ (Slide 2).

2. Model Evaluation & Theory (20 Marks)

Question: Explain why predictive accuracy alone can be misleading for imbalanced datasets. **Answer:** Predictive accuracy might overestimate the performance of a classifier in imbalanced scenarios. For example (Slide 6):

- **Scenario:** Predicting bankrupt companies where only 2% of companies are "bad."
- A classifier predicting all companies as "good" would achieve 98% accuracy but fail at identifying any "bad" companies. This highlights the need for more detailed measures such as **confusion matrix metrics** (Slide 8).

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: What is a confusion matrix, and how can it be used to evaluate model performance? **Answer:** A **confusion matrix** helps represent the results of testing in more detail:

- True Positive (TP): Correctly classified positive instances.
- False Positive (FP): Negative instances misclassified as positive.
- False Negative (FN): Positive instances misclassified as negative.
- True Negative (TN): Correctly classified negative instances.

Formula Examples (Slide 10):

- **Sensitivity:** $$\frac{\text{TP}}{\text{TP+FN}}$$.
- **Specificity:** $$\frac{\text{TN}}{\text{TN+FP}}$$.

4. Practical Insights (10 Marks)

Question: Discuss the trade-offs between false positives and false negatives in different applications. **Answer:** The impact depends on the domain:

- **Finance:** False positives are bad. Misclassifying "good" companies as "bad" can lead to missed investment opportunities (Slide 14).

- **Healthcare:** False negatives are bad. Misclassifying patients with conditions like brain tumors can delay diagnosis and treatment (Slide 16).

5. Computational Working & Analytical Understanding (20 Marks)

Question: Describe the ROC graph and explain its significance in evaluating classifiers.
Answer: An ROC graph plots:

- **TP Rate (Sensitivity):** Vertical axis.
- **FP Rate:** Horizontal axis.

Better classifiers are closer to the **top-left corner**, indicating high sensitivity and low false positives (Slide 22).

Based on the **Lecture 2** slide, here's how the exam questions and answers can be designed:

1. Conceptual Understanding & EDA (10 Marks)

Question: Define machine learning. Why is it important to learn from experience? **Answer:** Machine learning is defined as "any process by which a system improves performance from experience" (Slide 2). **Why is it important?** Learning enables systems to classify objects/events, solve problems, and achieve goals in dynamic environments. Examples from slides:

- Classification tasks like fraud detection in e-commerce and spam filtering in email (Slide 3).
- Problem-solving tasks like driving a car or controlling an elevator (Slide 4).

2. Model Evaluation & Theory (20 Marks)

Question: What are some performance metrics used to evaluate machine learning models? **Answer:** Performance metrics include:

1. **Classification Accuracy:** Percentage of correctly classified instances (Slide 5).
2. **Solution Correctness:** How accurate the solution is to a problem.
3. **Solution Quality:** Efficiency or optimization of the solution (e.g., shortest path).
4. **Speed of Performance:** The time required for computation or task completion.

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: Discuss the "Knowledge Engineering Bottleneck" and its implications for machine learning systems. **Answer:** The Knowledge Engineering Bottleneck occurs when systems require skills or knowledge that are too detailed or expensive to construct manually. Machine learning addresses this by enabling systems to adapt automatically or discover new knowledge from data. **Examples from slides:**

- Personalized systems like news filters (Slide 6).
- Data mining for insights, e.g., market basket analysis (Slide 7).

4. Practical Insights (10 Marks)

Question: Provide real-world applications where machine learning plays a crucial role. **Answer:** Examples of real-world applications:

- Financial investments: Predicting stock prices.
- DNA sequencing: Categorizing sequences.
- Spam filtering: Classifying emails as spam or legitimate (Slide 3).

5. Computational Working & Analytical Understanding (20 Marks)

Question: Explain the components of a learning system with examples. **Answer:** A learning system consists of:

1. **Training Experience:** E.g., observing human driver commands (Slide 10).
2. **Target Function:** What the system learns, e.g., categorizing emails as spam or legitimate.
3. **Performance Metric:** E.g., percentage of correct classifications.
4. **Learning Algorithm:** The method to infer the target function.

Alright, Syed! Based on **Lecture 3**, which covers **Inductive Classification**, here's how exam questions can be designed:

1. Conceptual Understanding & EDA (10 Marks)

Question: Define classification and categorization. What distinguishes a binary function from general categorization? **Answer:**

- **Classification (Categorization):** Assigns each instance xx (from instance space XX) to a fixed set of categories C={c1,c2,...cn}C = \{c_1, c_2, ... c_n\}. Example: Categorizing fruits into apples, oranges, etc. (Slide 2).
- **Binary Function:** Special case where C={0,1}C = \{0, 1\}, used for concepts like True/False or Positive/Negative (Slide 2).

2. Model Evaluation & Theory (20 Marks)

Question: Explain hypothesis selection and its importance in classification. **Answer:**

- **Hypotheses:** Potential functions h(x)h(x) that approximate the categorization function c(x)c(x).
- **Selection Criteria:** Besides consistency with training data, bias influences selection. *Example*:
  - Hypothesis: h(x)=h(x) = red & circle → positive.
  - Bias: Simple hypotheses generalize better (Occam's Razor) (Slide 6).

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: Differentiate between generalization and memorization in classification models. **Answer:**

- **Memorization:** Remembers training data examples but fails on unseen data.
- **Generalization:** Learns rules to classify unseen examples accurately. Example: Occam's Razor recommends simpler hypotheses for better generalization (Slide 7).

4. Practical Insights (10 Marks)

Question: Describe a real-world application of inductive classification. **Answer:** Example:

1. Spam Filtering: Classify emails as spam or legitimate (Slide 6).
2. Market Basket Analysis: Predict customer buying behavior based on transaction data (Slide 4).

5. Computational Working & Analytical Understanding (20 Marks)

Question: Explain the inductive learning hypothesis and its assumptions. **Answer:**

- **Hypothesis:** Functions approximating the target concept on large training sets will perform well on unseen examples.

- **Assumption:** Training and test examples come from the same distribution. Example: Learning a concept like red & circle → positive (Slide 8).

Understood! Let's align the exam questions with **Lecture 3 - Classification** content. Here's how the questions can be framed:

1. Conceptual Understanding & EDA (10 Marks)

Question: Define supervised and unsupervised learning. Highlight key differences and applications. **Answer:**

- **Supervised Learning:** The training data includes labels indicating the class of the observations. New data is classified based on the training set (Slide 2). Example: Fraud detection, medical diagnosis.
- **Unsupervised Learning:** The class labels of the training data are unknown; the aim is to establish the existence of clusters or classes within the data (Slide 2). Example: Customer segmentation in marketing.

2. Model Evaluation & Theory (20 Marks)

Question: Explain the two-step process of classification using an example. **Answer:**

1. **Model Construction:** A classifier is built using a training set, represented as rules, decision trees, or formulas (Slide 4). Example: Training set:

```
2. IF rank = 'professor' OR years > 6 THEN tenured = 'yes'
```

3. **Model Usage:** This classifier is applied to unseen data to predict outcomes (Slide 6).

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: What is information gain, and why is it essential for decision tree construction? **Answer:**

- **Information Gain:** The reduction in entropy after a dataset split on an attribute. It is used to choose the best attribute for splitting in decision trees (Slide 9). Formula: $$ Gain(A) = Info(D) - Info_A(D) $$
- **Example (Slide 11):** Calculating entropy for splitting based on “age” attribute in a dataset.

4. Practical Insights (10 Marks)

Question: Discuss an enhancement to basic decision tree induction and its benefits. **Answer: Enhancements (Slide 19):**

1. Continuous-valued attributes: Dynamically define intervals for these attributes.
2. Handle missing values: Assign most common or probabilistic values.

Benefit: Reduces noise and increases robustness in classification.

5. Computational Working & Analytical Understanding (20 Marks)

Question: Explain overfitting in decision trees and methods to avoid it. **Answer:**

- **Overfitting:** Occurs when a tree is too complex, reflecting anomalies in training data rather than the general structure (Slide 18).
- **Avoidance Methods:**
  1. **Prepruning:** Stop tree construction early.
  2. **Postpruning:** Remove branches from a fully grown tree using a validation set.

Based on the **Supervised Learning** slide content, I’ve structured exam questions and answers to align with the material. Here’s how the topics can be covered:

1. Conceptual Understanding & EDA (10 Marks)

Question: Define supervised learning and explain its purpose with an example. **Answer:** Supervised learning is a method where a model is trained on input-output pairs to learn a mapping from inputs to outputs. The purpose is to generalize this mapping to unseen data (Slide 3).

Example:

- Inputs: Age and mileage of a used car (e.g., Toyota Prius).
- Output: Predicted price of the car. This demonstrates the model's ability to predict car prices from given attributes.

2. Model Evaluation & Theory (20 Marks)

Question: What is the purpose of a loss function in supervised learning? Provide an example. **Answer:** A loss function measures how well the model's predicted outputs align with the actual outputs. It quantifies the error to optimize the model's performance during training (Slide 8).

Example: For linear regression, the least squares loss function minimizes the squared differences between predicted and actual values: $$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$ Where:

- yiy_i: Actual value.
- y^i\hat{y}_i: Predicted value.

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: Explain the importance of testing in supervised learning. **Answer:** Testing evaluates how well a trained model generalizes to new, unseen data. It helps identify issues like overfitting, where the model performs well on training data but poorly on test data (Slide 13).

Example Metrics: Accuracy, Mean Squared Error (MSE), and R-squared measure the model’s performance on test datasets.

4. Practical Insights (10 Marks)

Question: Why is gradient descent used in training models like linear regression? **Answer:** Gradient descent iteratively adjusts model parameters to minimize the loss function, especially when a closed-form solution is not possible due to:

- Non-linear models (e.g., deep learning).
- Regularization penalties.
- High-dimensional optimization (Slide 25).

This allows models to converge towards optimal solutions effectively.

5. Computational Working & Analytical Understanding (20 Marks)

Question: Describe 1D linear regression and its components. Provide an example. **Answer:** 1D linear regression predicts a single output from a single input based on:

1. **Model:** $$y = mx + c$$
   - mm: Slope (parameter).
   - cc: Intercept (parameter).
2. **Loss Function:** Least squares measures the error: $$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$
3. **Training:** Gradient descent minimizes LL to find optimal mm and cc.
4. **Testing:** Checks how well the model generalizes on unseen data.

Example: Train the model on:

Input (X)=[1,2,3]Output (Y)=[2,4,6]\text{Input (X)} = [1, 2, 3] \quad \text{Output (Y)} = [2, 4, 6]

Find m≈2m \approx 2 and c≈0c \approx 0.

Got it, Syed! Based on the **Shallow Neural Networks** slides, here’s how the exam questions and answers can be crafted to align with the lecture content:

1. Conceptual Understanding & EDA (10 Marks)

Question: Explain the concept of shallow neural networks and their advantages over simple linear regression. **Answer:** Shallow neural networks are fully connected neural networks with a single hidden layer. They can model complex input-output mappings and are flexible enough to describe nonlinear relationships (Slide: Shallow neural networks).

Advantages over Linear Regression:

1. Linear regression is limited to straight-line mappings.
2. Shallow networks can handle multiple inputs, outputs, and describe arbitrarily complex relationships.
3. Universal approximation theorem: With enough hidden units, a shallow neural network can approximate any continuous function with desired precision.

2. Model Evaluation & Theory (20 Marks)

Question: What is the universal approximation theorem, and why is it significant for neural networks? **Answer:** The **universal approximation theorem** states that a shallow neural network with enough hidden units can approximate any continuous function on a compact subset of Rn\mathbb{R}^n to arbitrary precision (Slide: Universal approximation theorem).

Significance:

1. Demonstrates that shallow networks are highly expressive models.
2. Justifies using neural networks for diverse applications, from regression to classification.
3. Explains why neural networks are capable of modeling complex systems and processes.

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: Describe the role of activation functions in shallow neural networks. Provide examples. **Answer:** Activation functions introduce nonlinearity into neural networks, enabling them to approximate complex mappings (Slide: Activation function).

Examples:

1. **ReLU (Rectified Linear Unit):** Defined as f(x)=max(0,x)f(x) = \max(0, x), ReLU converts negative values to zero while retaining positive values. Use: Common in modern networks due to efficiency and reduced vanishing gradient issues.
2. **Other Functions:** Sigmoid, Tanh, etc.

ReLU allows shallow networks to represent piecewise linear functions, making them suitable for nonlinear tasks (Slide: Example shallow network).

4. Practical Insights (10 Marks)

Question: Explain the concept of hidden units and their role in neural networks. **Answer: Hidden Units (Slide: Hidden units):** Intermediate nodes in a network's hidden layer that transform inputs into activations using linear functions followed by an activation function.

Role:

1. Capture patterns or features in data.
2. Enable networks to model nonlinear relationships (e.g., compute piecewise linear functions).
3. Each hidden unit adds to the network's capacity to represent data intricacies.

5. Computational Working & Analytical Understanding (20 Marks)

Question: How many parameters does a shallow network with HH hidden units, II inputs, and OO outputs have? Explain the calculation. **Answer:** Total parameters: (I×H)+H+(H×O)+O(I \times H) + H + (H \times O) + O (Slide: Nomenclature).

1. **Input-to-Hidden Weights:** I×HI \times H.
2. **Hidden Layer Biases:** HH.
3. **Hidden-to-Output Weights:** H×OH \times O.
4. **Output Biases:** OO.

Example: For 33 inputs, 44 hidden units, 22 outputs:

- Parameters = (3×4)+4+(4×2)+2=12+4+8+2=26(3 \times 4) + 4 + (4 \times 2) + 2 = 12 + 4 + 8 + 2 = 26.

Based on the content of the **Deep Neural Networks** slides, here are exam questions tailored to the lecture material:

1. Conceptual Understanding & EDA (10 Marks)

Question: What are deep neural networks, and how are they different from shallow neural networks? **Answer:** Deep neural networks consist of **more than one hidden layer** (Slide 1). While shallow networks often work well for simple tasks, deep networks excel in:

1. **Complexity:** They can model intricate relationships by stacking layers.
2. **Applications:** Best suited for tasks in **computer vision**, **natural language processing**, and **graph neural networks** (Slide 54).

2. Model Evaluation & Theory (20 Marks)

Question: Explain hyperparameters in deep networks and their significance. **Answer:** Hyperparameters are fixed before training and include:

1. **Depth (K):** Number of layers in the network.
2. **Width (**nHn_H**):** Number of hidden units per layer (Slide 42).

Significance:

1. Optimize network performance.
2. Balances fitting (accuracy) and generalization.

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: Compare the fitting and generalization capabilities of shallow and deep networks. **Answer:** Deep networks:

- Easier fitting up to ~20 layers; needs advanced techniques for deeper models (Slide 62).
- Good generalization due to capturing patterns over larger datasets.

Shallow networks:

- Limited ability to model complex functions.
- Struggles with tasks requiring intricate relationships (Slide 54).

4. Practical Insights (10 Marks)

Question: Why are convolutional layers used in deep networks for images? **Answer:** **Convolutional layers** leverage:

1. **Local weights:** Operate on parts of the image.
2. **Shared weights:** Reuse across the image.
3. **Integration:** Gradually combine information across layers (Slide 61).

This reduces complexity and increases efficiency in modeling large inputs like images.

5. Computational Working & Analytical Understanding (20 Marks)

Question: How do deep networks achieve depth efficiency? **Answer:** Deep networks approximate functions more efficiently compared to shallow networks, requiring exponentially fewer hidden units for equivalent results (Slide 60). **Example:** For specific functions, deep networks significantly reduce computational cost and parameter count.

Based on the **Backpropagation** slide content, here is how the exam questions and answers can be structured:

1. Conceptual Understanding & EDA (10 Marks)

Question: What is the main purpose of the backpropagation algorithm in neural networks? **Answer:** The **backpropagation algorithm** is used to train multilayer neural networks by minimizing the error between the predicted outputs and the target outputs. It adjusts weights and biases iteratively using the gradient descent method to minimize the performance index, typically the **mean square error (MSE)** (Slide 8). This enables the network to approximate functions with high accuracy.

2. Model Evaluation & Theory (20 Marks)

Question: Explain how the chain rule is used in backpropagation to compute gradients. **Answer:** The chain rule of calculus is applied to calculate the partial derivatives of the error with respect to the weights in the hidden layers. Since the error is an indirect function of these weights, the chain rule helps propagate the error back through the layers to compute sensitivities. For example (Slide 10): If ff is a function of nn, which is a function of ww, then the derivative with respect to ww is: $$ \frac{\partial f}{\partial w} = \frac{\partial f}{\partial n} \cdot \frac{\partial n}{\partial w} $$

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: What are the differences between stochastic gradient descent (SGD) and batch training in backpropagation? **Answer:**

- **Stochastic Gradient Descent (SGD):** Updates weights after each training example is presented. *Advantage:* Faster updates, especially with large datasets. *Disadvantage:* Noisy gradients can lead to slower convergence.
- **Batch Training:** Computes the gradient using the entire training dataset before updating weights. *Advantage:* More stable gradients. *Disadvantage:* Computationally expensive for large datasets. (Slide 30).

4. Practical Insights (10 Marks)

Question: How does the architecture of a network influence its ability to approximate functions? **Answer:** The architecture, such as the number of neurons and layers, determines the network's complexity and its capacity to approximate functions. For instance:

- A 1-3-1 network with sigmoid neurons in the hidden layer can approximate functions with three inflection points, but increasing complexity (e.g., i>4i > 4) requires more neurons in the hidden layer (Slide 36).

5. Computational Working & Analytical Understanding (20 Marks)

Question: Demonstrate the process of backpropagation for a simple 1-2-1 network approximating a function. **Answer:**

1. **Initialization:** Start with small random weights and biases (Slide 22).
2. **Forward Propagation:** Compute network output for a given input.
3. **Error Calculation:** Compute the error between output and target values (Slide 25).
4. **Backpropagation:** Use sensitivities and the chain rule to adjust weights by propagating the error back (Slide 15).
5. **Weight Update:** Apply the steepest descent method to update weights until error converges (Slide 28).

Example: For pp in [-2, 2], sample 21 points and approximate the sine function (Slide 23).

Based on the **Support Vector Machines (SVM)** lecture content, here is how exam questions can be aligned with the material:

1. Conceptual Understanding & EDA (10 Marks)

Question: What are Support Vector Machines (SVMs), and how do they achieve good generalization? **Answer:** SVMs are a supervised learning algorithm that finds a hyperplane separating two classes in high-dimensional feature space with the largest margin (Slides: Support Vector Machines). **Key Features:**

1. Large-margin hyperplanes ensure better generalization and prevent overfitting (Slide 18).
2. SVMs use "support vectors," which are critical datapoints that influence the decision boundary (Slide 21).

2. Model Evaluation & Theory (20 Marks)

Question: Explain the VC dimension and its role in structural risk minimization. **Answer:** The **VC dimension** measures the capacity of a model class, i.e., the number of datapoints a model can "shatter" (Slide 10). **Significance:** Lower VC dimension reduces the gap between training and testing error, which aids structural risk minimization (Slide 14). **Example:**

- VC dimension of hyperplane in 2D = 3.
- VC dimension in k dimensions = k+1k+1.

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: Describe the kernel trick and how it enables efficient computation in SVMs. **Answer:** The **kernel trick** computes scalar products in high-dimensional feature space using mappings from low-dimensional inputs without explicitly calculating the high-dimensional vectors (Slide 28). **Example Kernels:**

1. Polynomial kernel.
2. Gaussian radial basis function kernel (Slide 35).

Benefit: Makes finding maximum-margin hyperplanes computationally feasible for large feature spaces.

4. Practical Insights (10 Marks)

Question: What happens if classes are not linearly separable? **Answer:** For non-linear separability:

1. Introduce slack variables to allow planes closer to datapoints (Slide 50).
2. Use larger feature spaces for linear separability (Slide 44).
3. Apply kernel methods to create curved decision boundaries (Slide 57).

5. Computational Working & Analytical Understanding (20 Marks)

Question: Define the optimization problem for finding the maximum-margin separator.
Answer: SVM optimization minimizes: $$ \frac{1}{2} ||w||^2 $$ Subject to:

1. w·xi+b≥1w \cdot x_i + b \geq 1 for positive cases.
2. w·xi+b≤−1w \cdot x_i + b \leq -1 for negative cases. (Slide 24).

Solution: Quadratic programming solves the convex problem efficiently, allowing precise determination of the separating plane (Slide 25).

Based on the **Principal Component Analysis (PCA)** lecture content, here is how exam questions can be tailored to reflect the key concepts from the slides:

1. Conceptual Understanding & EDA (10 Marks)

Question: Explain the purpose of Principal Component Analysis (PCA) and its key applications. **Answer:** PCA is a dimensionality reduction algorithm that finds orthogonal directions of maximum variance in the data. By projecting data along these directions, PCA reduces dimensionality while retaining the most significant information (Slide 4). **Applications:**

- Visualization.
- Feature extraction and engineering.
- Reducing redundant/correlated features (Slide 5).

2. Model Evaluation & Theory (20 Marks)

Question: Why is removing redundant or correlated features essential, and how does PCA help achieve this? **Answer: Reasons to remove correlated features (Slide 10):**

1. Reduces slow convergence during optimization.
2. Improves interpretability of models.
3. Mitigates the curse of dimensionality.

How PCA helps:

- PCA identifies the principal axes of maximum variance, effectively reducing redundancy by projecting data onto these axes while retaining most of the information (Slide 12).

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: What is the role of variance in PCA? Explain with an example. **Answer:** Variance measures the information content in a variable; higher variance implies higher information. PCA projects data onto axes where the variance is maximized (Slide 9). **Example (Slide 8):** In a dataset with two variables (x1,x2x_1, x_2) that are linearly correlated, PCA reduces dimensionality by retaining the axis with maximum variance, minimizing information loss.

4. Practical Insights (10 Marks)

Question: Highlight the significance of eigenvectors and eigenvalues in PCA. **Answer:** **Eigenvectors:** Represent the principal axes of the data. **Eigenvalues:** Quantify the variance captured by each principal component (Slide 14). **Significance:**

- Eigenvectors determine the directions of maximum variance.
- Eigenvalues indicate the magnitude of variance along these directions. By projecting data onto eigenvectors, PCA minimizes information loss while reducing dimensionality (Slide 12).

5. Computational Working & Analytical Understanding (20 Marks)

Question: Outline the steps involved in PCA and describe how dimensionality reduction is achieved. **Answer:** Steps in PCA (Slide 36):

1. Compute the covariance matrix of the data.
2. Find the eigenvectors (principal axes) and eigenvalues of the covariance matrix.
3. Sort the eigenvectors by eigenvalues in descending order.
4. Select the top kk eigenvectors and project the data onto these components to reduce dimensionality.

Dimensionality Reduction: By selecting fewer principal components than the original dimensions, PCA reduces data size while preserving most variance (Slide 36).

Based on the content of the **PCA and Clustering** slides, here's how we can design exam questions aligned with the material:

1. Conceptual Understanding & EDA (10 Marks)

Question: Explain the purpose of Principal Component Analysis (PCA) and its application in dimensionality reduction. **Answer:** PCA reduces high-dimensional data by projecting it onto directions (principal components) that maximize variance, preserving as much information as possible (Slide: PCA Introduction). **Application:**

- Dimensionality reduction for visualization.
- Feature extraction to remove redundant or correlated features (Slide: Why PCA?).

2. Model Evaluation & Theory (20 Marks)

Question: Discuss the mathematical formulation of PCA and its connection to eigenvectors and eigenvalues. **Answer:** PCA identifies the eigenvectors (directions of maximum variance) of the covariance matrix.

1. Compute the covariance matrix of the data.
2. Find its eigenvalues and eigenvectors.
3. The principal components correspond to the eigenvectors with the largest eigenvalues (Slide: Mathematical Formulation).

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Question: What are the limitations of PCA, and when should it be avoided? **Answer:** **Limitations:**

- PCA transforms data into a new feature space, losing the interpretability of the original features.
- Dimensionality reduction using PCA may make classification tasks harder, especially if the problem is nonlinear (Slides: PCA Limitations).

- Linear assumptions: PCA assumes the data lies on a linear subspace, which may not hold true for complex, nonlinear relationships.

Alternatives:

- Use Kernel PCA for nonlinear relationships.

4. Practical Insights (10 Marks)

Question: What is clustering, and how does it differ from supervised learning? **Answer:** Clustering groups similar data samples into clusters based on a similarity measure. Unlike supervised learning, clustering does not require labeled examples and is a form of unsupervised learning (Slide: Clustering Overview).

Applications:

- Document clustering for information retrieval.
- Customer segmentation in marketing (Slide: Applications).

5. Computational Working & Analytical Understanding (20 Marks)

Question: Explain the k-means clustering algorithm. Why is the choice of initial centroids important? **Answer: Steps in k-means:**

1. Initialize kk centroids randomly.
2. Assign points to the nearest centroid to form clusters.
3. Update centroids by taking the mean of the points in each cluster.
4. Repeat until centroids stabilize (Slide: K-means Clustering).

Importance of Initial Centroids: Poor initialization can lead to suboptimal clusters, affecting the quality of results and convergence speed (Slides: Importance of Choosing Initial Centroids).

1. Conceptual Understanding & Exploratory Data Analysis (EDA) (10 Marks)

Key ML Concepts:

- **Supervised Learning:**
  Learn a function f(·) from labeled data.
  Example: Predict house prices using features like size and location.
- **Unsupervised Learning:**
  Identify patterns or groupings in data without labels.
  Example: Cluster customers based on purchase behavior.
- **Inductive Learning & Classification:**
  Learn general rules (or hypotheses) from examples. Techniques like decision trees use "if–then" rules to classify data.

Exploratory Data Analysis (EDA):

- **Missing Values:**
  Check data using methods like `isnull().sum()` and handle missing entries (imputation or drop).
- **Visualization:**
  - *Class Distribution:* Bar plots or count plots show the number of instances per class.
  - *Correlation Analysis:* Heatmaps visualize relationships among variables.
  - *Outlier Detection:* Box plots reveal extreme values.
- **Normalization and Splitting:**
  Scale features (e.g., using StandardScaler) and split the data into training and testing subsets.

Numerical Example & Python Code for EDA:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Load dataset
data = pd.read_csv('data.csv')
print("Missing values:\n", data.isnull().sum())

# Drop unneeded columns (example: 'id', 'Unnamed: 32')
data = data.drop(['id', 'Unnamed: 32'], axis=1)

# Visualize class distribution (assuming target column is 'diagnosis')
sns.countplot(x='diagnosis', data=data)
plt.title("Diagnosis Distribution")
plt.show()

# Correlation heatmap
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()

# Outlier detection: Box plot for a feature (e.g., 'radius_mean')
sns.boxplot(x=data['radius_mean'])
plt.title("Box Plot: radius_mean")
plt.show()

# Normalize features
scaler = StandardScaler()
num_features = data.select_dtypes(include=['float64', 'int64']).columns
data[num_features] = scaler.fit_transform(data[num_features])

# Split data: assuming 'diagnosis' is target
X = data.drop('diagnosis', axis=1)
y = data['diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, 
random_state=42)
print("Train shape:", X_train.shape, "Test shape:", X_test.shape)
```

---

2. Model Evaluation & Theory (20 Marks)

Optimization Methods:

- **Gradient Descent:**
  Iterative update to minimize loss.
  Mathematical Formulation:

  θ=θ−α·∇J(θ)\theta = \theta - \alpha \cdot \nabla J(\theta)

  where α\alpha is the learning rate.
  Numerical Example: Fitting a linear relation like y=2xy=2x.

- **Backpropagation:**
  Used in neural networks to compute gradients via the chain rule. It updates weights in each layer:

  W(l)=W(l)−α·∂J∂W(l)W^{(l)} = W^{(l)} - \alpha \cdot \frac{\partial J}{\partial W^{(l)}}

Model Theories:

- **Support Vector Machines (SVM):**
  Find a maximum-margin hyperplane; the kernel trick maps data to higher dimensions to handle non-linear separability.
- **Deep Neural Networks:**
  Use multiple hidden layers; trained via gradient descent with backpropagation.
- **Decision Trees:**
  Use heuristics like information gain (or Gini) for splitting.

Python Code Example for Gradient Descent:

```
import numpy as np

# Example: Linear regression with gradient descent on points (1,2), (2,4),
(3,6), (4,8)
X = np.array([1, 2, 3, 4])
y = np.array([2, 4, 6, 8])

slope, intercept = 0.0, 0.0
learning_rate = 0.01
iterations = 1000

for i in range(iterations):
    y_pred = slope * X + intercept
    error = y - y_pred
    loss = np.mean(error**2)
    slope_grad = -2 * np.mean(error * X)
    intercept_grad = -2 * np.mean(error)

    slope -= learning_rate * slope_grad
    intercept -= learning_rate * intercept_grad

print("Final Slope:", slope, "Final Intercept:", intercept)
```

Python Code Example for SVM:

```
from sklearn.svm import SVC

# A simple SVM example on a tiny dataset
X_svm = [[0, 0], [1, 1]]
y_svm = [0, 1]
svm_model = SVC(kernel='linear', probability=True, random_state=42)
svm_model.fit(X_svm, y_svm)
print("Support Vectors:", svm_model.support_vectors_)
```

---

3. Evaluation Metrics and Their Uses in Different Scenarios (20 Marks)

Key Metrics:

- **Confusion Matrix:**
  Breaks down prediction results:
    - TP (True Positives)
    - FP (False Positives)
    - TN (True Negatives)
    - FN (False Negatives)
- **Precision:**
  Precision=TPTP+FP\text{Precision} = \frac{TP}{TP+FP}
  Use: When false positives are costly (e.g., spam filters).
- **Recall:**
  Recall=TPTP+FN\text{Recall} = \frac{TP}{TP+FN}
  Use: When false negatives are critical (e.g., disease detection).
- **F1-Score:**
  Harmonic mean of precision and recall.

- **ROC Curve & AUC:**
  ROC plots TPR vs. FPR; AUC summarizes overall performance.
- **Regression Metrics:**
  - Mean Squared Error (MSE): 1n∑(y−y^)2\frac{1}{n}\sum (y - \hat{y})^2
  - Mean Absolute Error (MAE): 1n∑|y−y^|\frac{1}{n}\sum |y - \hat{y}|

Python Code Examples:

```
from sklearn.metrics import confusion_matrix, precision_score,
recall_score, f1_score, roc_curve, auc

# Confusion Matrix, Precision, Recall, F1-Score
y_true = [0, 1, 1, 0, 1]
y_pred = [0, 1, 0, 0, 1]
cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:\n", cm)
print("Precision:", precision_score(y_true, y_pred))
print("Recall:", recall_score(y_true, y_pred))
print("F1-Score:", f1_score(y_true, y_pred))

# ROC Curve and AUC Example
y_scores = [0.1, 0.4, 0.35, 0.8]
fpr, tpr, thresholds = roc_curve([0, 0, 1, 1], y_scores)
roc_auc = auc(fpr, tpr)
print("AUC:", roc_auc)
```

---

4. Practical Insights (10 Marks)

Real-World Considerations:

- **Impact of PCA:**
  Dimensionality reduction via PCA can retain (e.g., 95%) of the data's variance while reducing feature complexity. In many cases, the performance difference is minimal if the original features already capture key information.
- **Model Trade-Offs:**
  - **Decision Trees:**
    Highly interpretable; use pruning to prevent overfitting.
  - **SVM:**
    Works well in high-dimensional spaces; kernel selection is crucial.
  - **Neural Networks:**
    Can learn complex non-linear relationships but require careful training and large datasets.
- **Medical Diagnosis Example:**
  In sensitive domains like medical diagnosis, false negatives (missing a disease) are critical. Thus, models with high recall (and a good F1-score) are preferred even if overall accuracy is similar.

---

5. Computational Working & Analytical Understanding of Models (20 Marks)

In-Depth Computational Techniques:

- **Gradient Descent & Backpropagation:**
  - *Mathematical Insight:*
    Update rule:

    θ=θ−α·∇J(θ)\theta = \theta - \alpha \cdot \nabla J(\theta)

    In neural networks, backpropagation applies the chain rule:

    ∂J∂Wl=sl·al−1\frac{\partial J}{\partial W^l} = s^l \cdot a^{l-1}

    where sls^l are the error sensitivities.
  - *Python Example:* (Shown above in the Gradient Descent code snippet.)
- **PCA Workflow:**
  - Steps:
    1. Normalize the dataset.
    2. Compute the covariance matrix.
    3. Find eigenvalues and eigenvectors.
    4. Order eigenvectors by descending eigenvalues.
    5. Project the data onto the top components.
  - *Python Example:*

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Sample dataset (2D example)
X_sample = np.array([
    [2.5, 2.4],
    [0.5, 0.7],
    [2.2, 2.9],
    [1.9, 2.2],
    [3.1, 3.0],
    [2.3, 2.7],
    [2.0, 1.6],
    [1.0, 1.1],
    [1.5, 1.6],
    [1.1, 0.9]
])

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_sample)
pca = PCA(n_components=0.95)  # Retains 95% of variance
X_pca = pca.fit_transform(X_scaled)
print("Explained Variance Ratio:", pca.explained_variance_ratio_)
print("Reduced Data Shape:", X_pca.shape)
```

- **SVM and Kernel Methods:**
  - *Mathematical Formulation:*
    SVM seeks the hyperplane that maximizes the margin, formulated as:

    max⁡w,bmin⁡i∥wTxi+b∥\max_{w,b} \min_i \|w^T x_i + b\|

    subject to yi(wTxi+b)≥1y_i (w^T x_i + b) \geq 1.

- *Python Example:* (Provided above under SVM section.)
- **Decision Trees: Information Gain:**
  - *Entropy Calculation:*

    H(X)=−∑ipilog⁡2(pi)H(X) = -\sum_{i} p_i \log_2(p_i)

    Example: For a binary split with p(positive)=0.6p(positive)=0.6 and p(negative)=0.4p(negative)=0.4:

    H(X)≈−(0.6log⁡20.6+0.4log⁡20.4)≈0.971H(X) \approx - (0.6 \log_2 0.6 + 0.4 \log_2 0.4) \approx 0.971

  - *Information Gain:*

    Gain(A)=H(X)−∑v∈Values(A)|Dv||D|H(Dv)\text{Gain}(A) = H(X) - \sum_{v \in \text{Values}(A)} \frac{|D_v|}{|D|} H(D_v)

  - *Python Example:*

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt

# Example dataset for a decision tree
X_dt = [[0, 0], [1, 1], [0, 1], [1, 0]]
y_dt = [0, 1, 0, 1]

clf_dt = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf_dt = clf_dt.fit(X_dt, y_dt)
plt.figure(figsize=(6,4))
tree.plot_tree(clf_dt, filled=True)
plt.title("Decision Tree (using Entropy)")
plt.show()
```

Final Thoughts:
These notes integrate theory, numerical examples, and practical code. They provide a framework to understand and implement:

- EDA and feature preprocessing,
- Core model evaluation techniques and the underlying theory (gradient descent, SVM, etc.),
- Detailed evaluation metrics (confusion matrix, ROC, etc.),
- Practical insights from model comparisons (e.g., PCA impact, model trade-offs),
- Computational working of models (mathematical formulations and code implementations).

Study these notes, run the provided code (modifying file paths and variable names as needed), and use the numerical examples to strengthen your understanding. Good luck with your exam preparation!