Machine Learning Exam Preparation

Sample Answers

(Splitting) Explain how k-fold cross-validation works

K-fold cross-validation is a technique used to assess the performance of a machine learning model by dividing the dataset into *k* equally sized subsets or "folds." The process involves the following steps:

1. Split the dataset into *k* folds.
2. For each fold, use it as a validation set and the remaining *k-1* folds as the training set.
3. Train the model on the training set and evaluate it on the validation set.
4. Repeat this process *k* times, each time with a different fold as the validation set.
5. Calculate the average performance across all *k* trials to get a more robust estimate of the model's performance.

This method helps in reducing overfitting and provides a better understanding of how the model will generalize to an independent dataset.

(Preprocessing) Explain how you would encode the race feature below and what would be the output of the encoder

To encode a categorical feature like "race," you can use techniques such as one-hot encoding or label encoding:

- **One-hot encoding:** This method creates binary columns for each category in the feature. For example, if "race" has categories like "White," "Black," and "Asian," one-hot encoding would create three new binary features: `race_White`, `race_Black`, and `race_Asian`. Each row will have a value of 1 in one of these columns and 0 in others.
- **Label encoding:** This method assigns an integer to each category. For example, "White" might be encoded as 0, "Black" as 1, and "Asian" as 2.

The choice between these methods depends on whether there is an ordinal relationship between categories (use label encoding) or not (use one-hot encoding).

(Evaluation metrics) Given the true and predicted target variables below, write down the confusion matrix and calculate the following metrics: accuracy, precision, recall. Make sure to label the axes of the matrix and write down the equations of the metrics using the names of the elements in the confusion matrix

Assuming binary classification with classes Positive (P) and Negative (N):

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

- **Accuracy**: $\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
- **Precision**: $\text{Precision} = \frac{TP}{TP + FP}$
- **Recall**: $\text{Recall} = \frac{TP}{TP + FN}$

These metrics provide insights into different aspects of model performance: accuracy gives overall correctness, precision measures correctness among positive predictions, and recall assesses coverage of actual positives.

(Evaluation metrics) Given the true target variable of a binary classification problem and the 2D array of predicted probabilities, write down the steps of how you calculate the ROC curve

To calculate an ROC curve:

1. Sort instances by predicted probability scores in descending order.
2. Set a threshold starting from 1 down to 0.
3. For each threshold:
    - Classify instances above threshold as positive; below as negative.
    - Calculate True Positive Rate (TPR = Recall) and False Positive Rate (FPR = FP / (FP + TN)).
4. Plot TPR against FPR at various thresholds to form an ROC curve.
5. The area under this curve is known as AUC, which indicates model performance.

(ML models) What is the mathematical model behind logistic regression? Write down the equation of how predictions are calculated based on feature values

Logistic regression uses a logistic function to model binary outcomes:

- The logistic function is defined as:

$h(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1x_1 + \beta_2x_2 + ... + \beta_nx_n)}}$

Where:

- $h(x)$ is the predicted probability that output is 1.
- $\beta_0, \beta_1, ..., \beta_n$ are parameters learned from data.
- $x_1, x_2, ..., x_n$ are input features.

Predictions are made by applying this function to input features.

(ML models) Explain how gradient descent works. Write down the equation of how weights are updated and the steps of the algorithm

Gradient descent is an optimization algorithm used to minimize a function by iteratively moving towards its steepest descent:

Steps:

1. Initialize weights randomly.
2. Calculate gradient of loss function with respect to weights.
3. Update weights using:

$w := w - \alpha \nabla L(w)$

Where:

- w are weights.
- α is learning rate.
- $\nabla L(w)$ is gradient of loss function.

4. Repeat until convergence or maximum iterations reached.

The goal is to find weights that minimize loss function.

Additional Questions

(Feature Engineering) Why is feature scaling important in machine learning models? How would you perform feature scaling?

Feature scaling ensures that all features contribute equally to distance calculations in algorithms like k-nearest neighbors or gradient descent optimization in models like neural networks:

- **Standardization (Z-score normalization):**

$x' = \frac{x - \mu}{\sigma}$

Where x' is scaled feature, x is original feature, μ is mean, σ is standard deviation.

- **Min-Max Scaling:**

$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$

Scales features to range [0, 1].

(Regularization) Explain L1 vs L2 regularization in linear models

Regularization helps prevent overfitting by adding penalty terms to loss functions:

- **L1 Regularization (Lasso):** Adds penalty equal to absolute value of coefficients:

$L = L_{\text{original}} + \lambda \sum |\beta_i|$

Encourages sparsity; some coefficients become zero.

- **L2 Regularization (Ridge):** Adds penalty equal to square of coefficients:

$L = L_{\text{original}} + \lambda \sum (\beta_i)^2$

Encourages small coefficients but not necessarily zero.

Both methods add complexity penalties but differ in their effect on model parameters.