🧾 Post-Session Notes: Revision – Neural Networks (NN) and Convolutional Neural Networks (CNNs)

🔍 1. Introduction and Project Overview

- **The session revisited neural network fundamentals within the context of a healthcare data project.**
- **The project used synthetic health-related features like sleep hours, calorie intake, heart rate, stress level, etc.**
- **The goal: predict well-being (e.g., "well-rested" or not) using classification models.**
- **The session marked the conclusion of Modules A & B, with emphasis on practical implementation and evaluation.**

🧪 2. Model Building Need & Data Analysis

- **Building models helps uncover hidden patterns and support healthcare decision-making.**
- **Effective data exploration is essential before model training:**
  - **Visualizing relationships (e.g., how stress relates to sleep).**
  - **Feature engineering to combine correlated variables (e.g., combining sleep quality and heart rate).**

- Strong warning: Never include target variables as features—this causes data leakage and gives falsely perfect models.

🧹 3. Data Preprocessing & Feature Selection

- Key steps:
    - Handling missing values appropriately.
    - Standardizing or normalizing input features.
    - Selecting only relevant variables to avoid overfitting or redundancy.
- Feature thresholds (e.g., a cutoff for “good sleep” at score > 70) were explored for improved classification clarity.

🧾 4. Understanding Evaluation Metrics

- A strong focus was placed on confusion matrix interpretation:
    - True Positives (TP)
    - False Positives (FP)
    - True Negatives (TN)
    - False Negatives (FN)

- **Other key metrics:**
    - **Precision: TP / (TP + FP)**
    - **Recall: TP / (TP + FN)**
    - **F1 Score: Harmonic mean of precision & recall**
    - **ROC Curve & AUC: For overall model discrimination ability**

Accuracy alone is misleading, especially with imbalanced datasets.

⚖️ 5. Handling Class Imbalance with Class Weights

- **In real-world healthcare datasets, imbalance is common (e.g., very few "not well-rested" cases).**
- **Class weights in models help focus learning on the minority class.**

Example:

LogisticRegression(class_weight='balanced')

-

🧮 6. Model Progression: From Logistic Regression to Deep Learning

🔸 **Logistic Regression**

- **Baseline model using sigmoid activation for binary classification.**

- Outputs probability between 0 and 1.

🔸 Perceptron

- Single-layer neural network with binary output.
- Composed of linear layer + activation (e.g., sigmoid or ReLU).
- Learns weights via gradient descent and backpropagation.

🔸 Deep Neural Networks (DNN)

- Multiple hidden layers stacked with nonlinear activations (e.g., ReLU).

Architecture example:

Input (5 features) → Dense(32) → ReLU → Dense(16) → ReLU → Output (1 node) → Sigmoid

-
- Showed incremental improvements over simpler models.

🔢 7. Activation Functions

- Sigmoid: Smoothly maps input to (0,1) — good for binary classification, but suffers from vanishing gradient.
- ReLU (Rectified Linear Unit): Most common in deep networks, avoids vanishing gradients.

🔄 8. Loss Functions & Optimizers

- Binary Cross-Entropy Loss (BCELoss): Used for binary classification.
- Adam Optimizer: Combines momentum and adaptive learning rates — faster convergence than SGD.

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

🔁 9. Training Neural Networks: Best Practices

- Training involves:
  - Forward pass → Calculate predictions
  - Loss computation
  - Backward pass → Use `.backward()` for gradient calculation
  - Weight updates via optimizer
- Epochs: Full passes through training data
- Overfitting risks addressed through:
  - Proper validation
  - Feature selection
  - Avoiding data leakage

⚠️ 10. Avoiding Data Leakage

- Do NOT include target/dependent variable in feature set.
- Leads to artificially high accuracy and non-generalizable models.
- Emphasized as a critical pitfall in modeling.

🤖 11. Use of LLMs for Summarizing Results

- LLMs like ChatGPT can help:
  - Summarize metrics
  - Compare models in a tabular format
  - Provide interpretation and recommendations
- Encouraged to use LLMs for efficient documentation and insight generation after multiple model experiments.

🧠 12. Convolutional Neural Networks (CNNs): When & Why

- CNNs are designed for:
  - Image data (e.g., 2D grids)

    - Time series with spatial/temporal structure
- For the current tabular healthcare data, CNNs are:
    - Inappropriate and unnecessary
    - Better suited models: DNNs or tree-based models

🔬 13. Cross-Validation & Multiple Model Runs

- Helps combat randomness from:
    - Weight initialization
    - Data shuffling
- Encouraged to:
    - Run each model multiple times
    - Use k-fold cross-validation to assess performance stability
    - Average metrics across folds

📊 14. Final Remarks & Recommendations

- Use multiple models (e.g., logistic regression, perceptron, DNN) and compare results.

- No model is universally best; trade-offs exist between precision, recall, and interpretability.
- Experimentation with architecture, hyperparameters, and features is key.
- Balance datasets or use class weights to ensure fairness.
- Avoid misuse of models by respecting data boundaries and evaluation rigor.
- Encourage continuous practice, especially using tools like Keras for rapid prototyping.

📌 15. Key Takeaways

Aspect	Best Practice
Evaluation	Use precision, recall, F1, confusion matrix
Class imbalance	Use class weights or balance the dataset
Feature selection	Avoid including target feature, engineer meaningfully
Model types	Start simple (logistic) → perceptron → DNN
CNNs	Only for spatial/temporal data (images, time series)
Tools	Use LLMs for result summaries; Keras for prototyping
Training	Experiment with layers, activations, optimizers

Validation	**Cross-validate and repeat runs for reliability**