DAY 7- DAY 12

1 What is Fine-Tuning in Machine Learning?

Fine-tuning is the process of improving a machine learning model's performance by adjusting certain components like hyperparameters, data preprocessing, or training on task-specific data. It generally involves:

- Selecting the best hyperparameters.
- Feature engineering to enhance model input.
- Data normalization/scaling.
- Applying model validation techniques.

2 Hyperparameter Tuning

Hyperparameters are the settings/configurations of the model that are set before training (like learning rate, depth of a tree, etc.).
Hyperparameter tuning is the process of searching for the best combination of hyperparameters to maximize the model's performance.

3 Grid Search vs Random Search

Aspect	Grid Search	Random Search
Definition	Tries **all combinations** of hyperparameters in a given grid.	Tries **random combinations** of hyperparameters from a specified range.
Use	For small search spaces with limited parameters.	For large search spaces where trying all combinations is not practical.
Why Needed	To systematically find the best parameters.	To save time and compute resources while still exploring parameter space.

☑ Example (Scikit-learn - GridSearchCV):

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
grid = GridSearchCV(SVC(), param_grid, cv=5)
grid.fit(X_train, y_train)
print("Best parameters:", grid.best_params_)
```

```
Best parameters: {'C': 1, 'kernel': 'rbf'}
```

☑ Example (Scikit-learn - RandomizedSearchCV):

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform

param_dist = {'C': uniform(0.1, 10), 'kernel': ['linear', 'rbf']}
rand_search = RandomizedSearchCV(SVC(), param_distributions=param_dist, n_iter=10, cv=5)
rand_search.fit(X_train, y_train)
print("Best parameters:", rand_search.best_params_)
```

```
Best parameters: {'C': np.float64(7.464754280491552), 'kernel': 'rbf'}
```

Why Use Random Search if Grid Search Seems Better?

- Grid Search **tests all options**, which becomes impractical with many parameters.
- Random Search **covers a wide range quickly** and may find good parameters faster.
- Research shows that Random Search is often **more efficient** in high-dimensional spaces.

4 Feature Engineering

Process of selecting, transforming, or creating features to improve model learning.
Examples:

- Removing irrelevant features.
- Creating interaction features.
- Encoding categorical variables.

5 Normalizing Data

- Ensures all features are on the same scale.
- Helps algorithms like SVM, KNN, and Gradient Descent converge faster.
- Common techniques: **Min-Max Scaling, Standardization (Z-Score)**.

6 Model Validation

Used to check model performance on unseen data and avoid overfitting.
Common methods:

- **Train/Test Split**
- **Cross-Validation (like K-Fold)**
- **Stratified K-Fold for imbalanced data**

7 Use of 4 Important Libraries in Hyperparameter Tuning

Library	Use
Scikit-Learn	Provides GridSearchCV, RandomizedSearchCV, pipelines.
Optuna	Automatic hyperparameter optimization using advanced search algorithms.
Hyperopt	Uses TPE (Tree-structured Parzen Estimator) algorithm for optimization.
Bayesian Optimization	Uses probabilistic models to select the next best hyperparameter combination.

8 What is Bayesian Optimization?

Bayesian Optimization is a sequential model-based optimization technique.

- It builds a **probabilistic model (surrogate)** of the objective function.
- Chooses the next parameters based on **expected improvement**.
- Suitable when each evaluation is costly (e.g., training deep models).

☑ **Example using bayes_opt:**

```
from bayes_opt import BayesianOptimization

def black_box_function(x, y):
    return -x ** 2 - (y - 1) ** 2 + 1

pbounds = {'x': (-2, 2), 'y': (-3, 3)}

optimizer = BayesianOptimization(f=black_box_function, pbounds=pbounds, random_state=1)
optimizer.maximize(init_points=2, n_iter=5)
print(optimizer.max)
```

```
|   iter    |  target   |     x     |     y     |
-------------------------------------------------
| 1         | 0.7861845 | -0.331911 | 1.3219469 |
| 2         | -7.776786 | -1.999542 | -1.186004 |
| 3         | -0.979189 | 0.1552669 | -0.398242 |
| 4         | 0.4507370 | 0.7411132 | 0.9962324 |
| 5         | -3.352503 | 0.5937200 | 3.0       |
| 6         | -5.510865 | 2.0       | -0.584571 |
| 7         | -2.677723 | -1.616426 | 2.0319337 |
=================================================
{'target': np.float64(0.7861845912690542), 'params': {'x': np.float64(-0.331911981189704), 'y': np.float64(1.3219469606529488)}}
```

9️⃣ What is Optuna? (Brief)

- Optuna is an automatic hyperparameter optimization framework.
- It uses **Define-by-Run API**, allowing dynamic search spaces.
- It supports pruning of unpromising trials.

☑ Example Optuna Usage:

```
import optuna

def objective(trial):
    x = trial.suggest_float('x', -10, 10)
    return (x - 2) ** 2

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=100)
print("Best params:", study.best_params)
```

```
e: 0.0011110259422668653.
[I 2025-07-15 23:21:28,957] Trial 92 finished with value: 0.6822581764699588 and parameters: {'x': 1.1740107891322316}. Best is trial 44 with value:
0.0011110259422668653.
[I 2025-07-15 23:21:28,971] Trial 93 finished with value: 4.566644328267271 and parameters: {'x': -0.1369708299915366}. Best is trial 44 with valu
e: 0.0011110259422668653.
[I 2025-07-15 23:21:28,979] Trial 94 finished with value: 0.016432579883678165 and parameters: {'x': 1.8718103752884885}. Best is trial 44 with valu
e: 0.0011110259422668653.
[I 2025-07-15 23:21:28,990] Trial 95 finished with value: 0.45338892575167716 and parameters: {'x': 2.6733416114808866}. Best is trial 44 with valu
e: 0.0011110259422668653.
[I 2025-07-15 23:21:29,029] Trial 96 finished with value: 9.449139101760345 and parameters: {'x': 5.073945201489503 5}. Best is trial 44 with value:
0.0011110259422668653.
[I 2025-07-15 23:21:29,040] Trial 97 finished with value: 1.561000542952761 and parameters: {'x': 3.249400073216246}. Best is trial 44 with value:
0.0011110259422668653.
[I 2025-07-15 23:21:29,045] Trial 98 finished with value: 1.5735681648850595 and parameters: {'x': 0.7455805466730598}. Best is trial 44 with value:
0.0011110259422668653.
[I 2025-07-15 23:21:29,057] Trial 99 finished with value: 0.007585998016610939 and parameters: {'x': 1.9129023650343424}. Best is trial 44 with valu
e: 0.0011110259422668653.
Best params: {'x': 1.9666679442238126}
```

🔟 PCA (Principal Component Analysis) — Explained Briefly

PCA is a dimensionality reduction technique.

- It finds new axes (principal components) which capture maximum variance.
- Helps reduce overfitting and speeds up training by reducing features.

☑ PCA Example:

```
import optuna

def objective(trial):
    x = trial.suggest_float('x', -10, 10)
    return (x - 2) ** 2

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=100)
print("Best params:", study.best_params)
```

```
e: 2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,210] Trial 92 finished with value: 0.467974147739567 and parameters: {'x': 1.3159136401450713}. Best is trial 71 with value:
2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,218] Trial 93 finished with value: 0.3508176595400154 and parameters: {'x': 2.592298623618201}. Best is trial 71 with value:
2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,229] Trial 94 finished with value: 1.6804921925658813 and parameters: {'x': 3.2963379931815164}. Best is trial 71 with value:
2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,239] Trial 95 finished with value: 6.94756755803039 and parameters: {'x': -0.6358238860042205}. Best is trial 71 with value:
2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,271] Trial 96 finished with value: 0.008521757556681301 and parameters: {'x': 2.092313366078165}. Best is trial 71 with valu
e: 2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,287] Trial 97 finished with value: 6.248791969013536 and parameters: {'x': 4.499758382126868}. Best is trial 71 with value:
2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,293] Trial 98 finished with value: 1.225569684406055 and parameters: {'x': 0.892945491673488}. Best is trial 71 with value:
2.0545998406670 44e-05.
[I 2025-07-15 23:24:19,302] Trial 99 finished with value: 0.019767428842842088 and parameters: {'x': 2.140596688591311}. Best is trial 71 with valu
e: 2.0545998406670 44e-05.
Best params: {'x': 2.004532769397032}
```

Summary Table:

Topic	Purpose
Grid Search	Exhaustive parameter search
Random Search	Efficient search for large spaces
Bayesian Optimization	Smart, probabilistic search
Optuna	Automated, flexible hyperparameter tuning
PCA	Dimensionality reduction