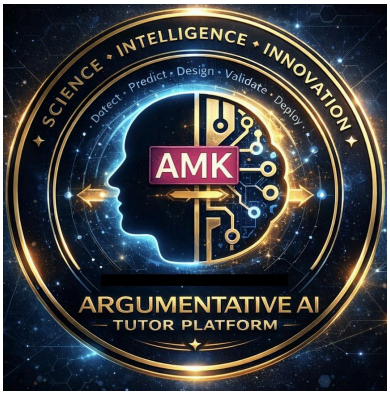
AMK RESEARCH LAB PROGRAM

Benchmark Lecture Note 16

Normalization, Principal Component Analysis (PCA), and Ensemble Models



AI Certification Program • AMK Research Lab Benchmark Series

Focus: preprocessing discipline, dimensionality reduction, robust model combination, and responsible deployment.

Why this matters: normalization makes features comparable, PCA compresses information into fewer dimensions, and ensemble models combine multiple learners to improve robustness. Together, these techniques sit at the heart of practical machine learning pipelines for health, finance, cybersecurity, manufacturing, and research automation.

Learning outcomes

- Explain why scaling and normalization affect optimization, distance-based methods, and feature comparability.
- Interpret PCA as an orthogonal projection that preserves as much variance as possible in fewer dimensions.
- Differentiate bagging, boosting, voting, and stacking as major families of ensemble learning.
- Design a simple pipeline in Python that combines preprocessing, dimensionality reduction, and model comparison.
- Relate these techniques to responsible AI through traceability, validation, and reproducible experimentation.

Pipeline overview

Figure 1. Typical workflow linking normalization, PCA, base models, and ensemble output.

1. Normalization and scaling

Many machine-learning algorithms are sensitive to feature scale. Gradient-based neural networks can converge slowly when inputs sit on very different numeric ranges, and distance-based models such as k-nearest neighbors or k-means can let one large-scale variable dominate the geometry of the problem.

Standardization is one common strategy. In practice, a feature x is often transformed to z = (x - mean) / standard deviation. Min–max scaling is another option when bounded intervals are desirable. The right choice depends on the distribution, model family, and operational context.

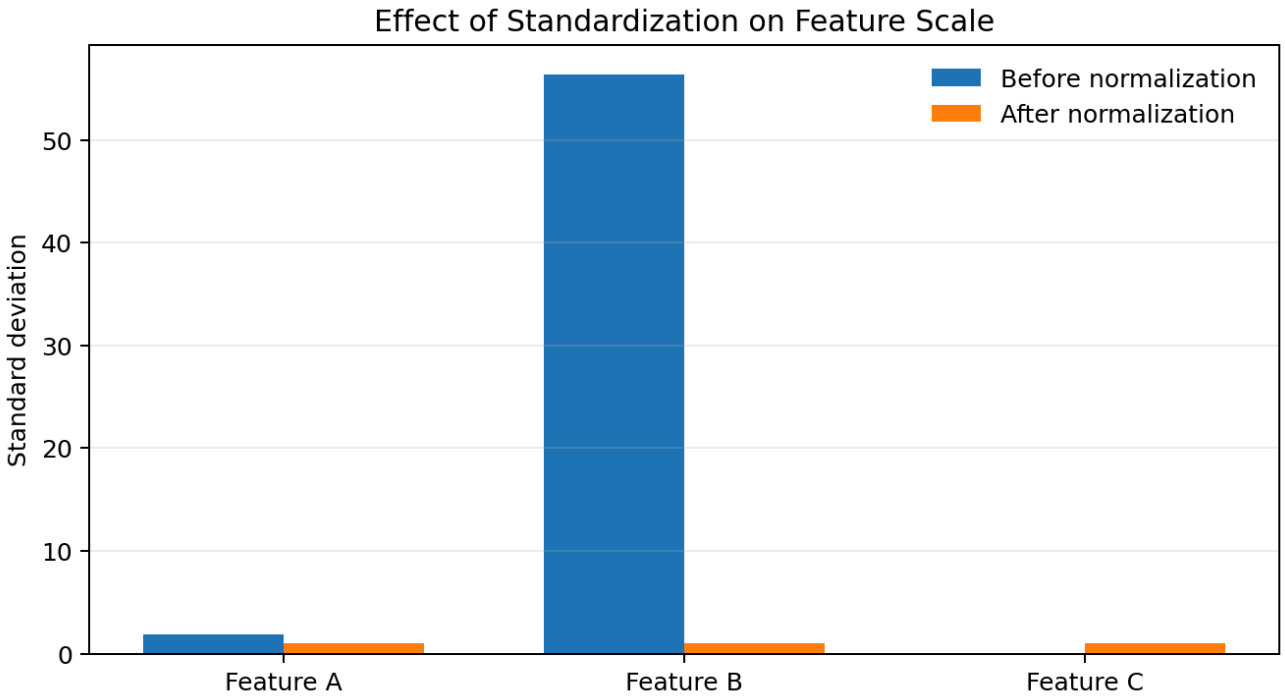


Figure 2. Standardization reduces scale imbalance across features.

Technique	Best use case	Key advantage	Watch-out
StandardScaler	Gradient methods; PCA; many linear models	Centers data and gives unit variance	Sensitive to outliers

MinMaxScaler	Neural nets with bounded input ranges	Maps data into a fixed interval	Can compress most values if outliers exist
Robust scaling	Heavy-tailed or outlier-rich data	Uses median and IQR, often more stable	May still need later standardization

2. Principal Component Analysis (PCA)

PCA is a linear dimensionality-reduction technique that rotates the coordinate system so the first component captures the largest possible variance, the second component captures the next largest variance orthogonal to the first, and so on.

In teaching practice, PCA is useful for compression, denoising, visualization, and feature engineering. It is not magic: components are linear combinations of the original variables, which can reduce interpretability if the class needs raw-feature explanations.

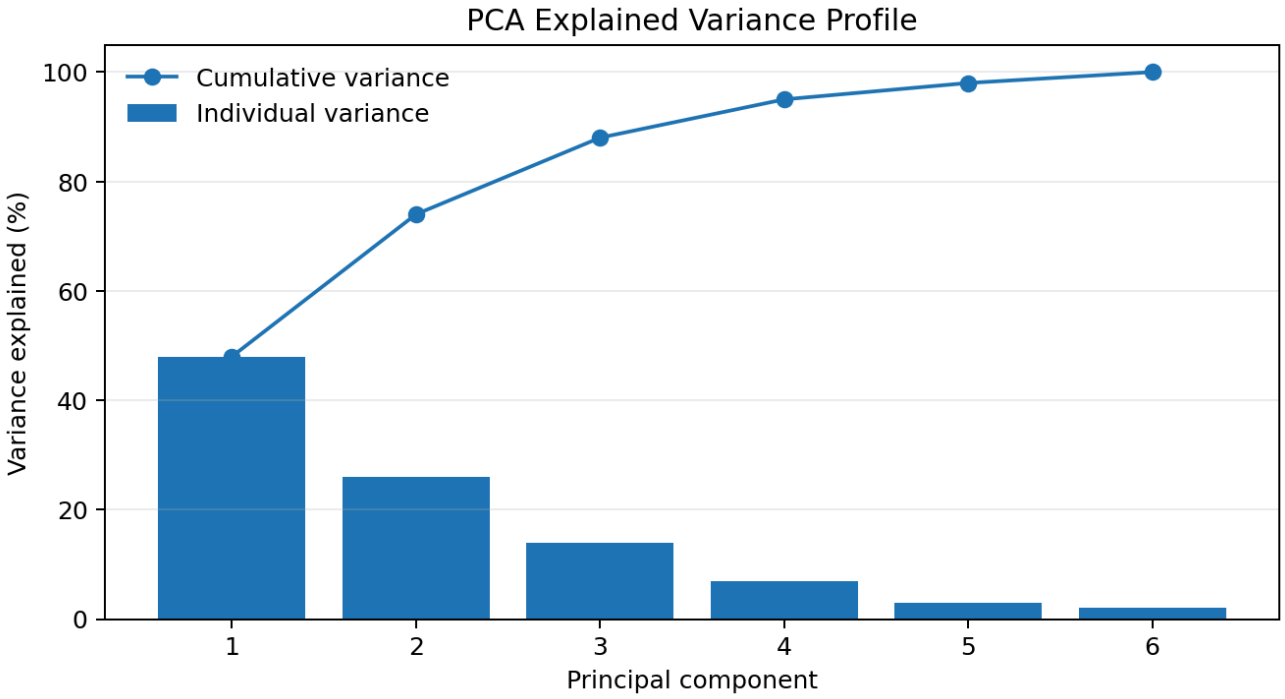


Figure 3. Example explained-variance profile for selecting the number of principal components.

Interpretation tip: select the smallest number of components that preserves the variance needed for the task while keeping the model auditable and computationally efficient.

3. Ensemble models

An ensemble combines multiple base estimators to improve generalization and robustness. Bagging reduces variance by training models on resampled data. Boosting builds models sequentially so later learners focus on earlier mistakes. Voting and stacking aggregate multiple predictors in parallel.

Ensemble methods are powerful because no single model family is universally best. In real projects, ensembles often improve stability across noisy data, imbalanced classes, and evolving operating conditions.

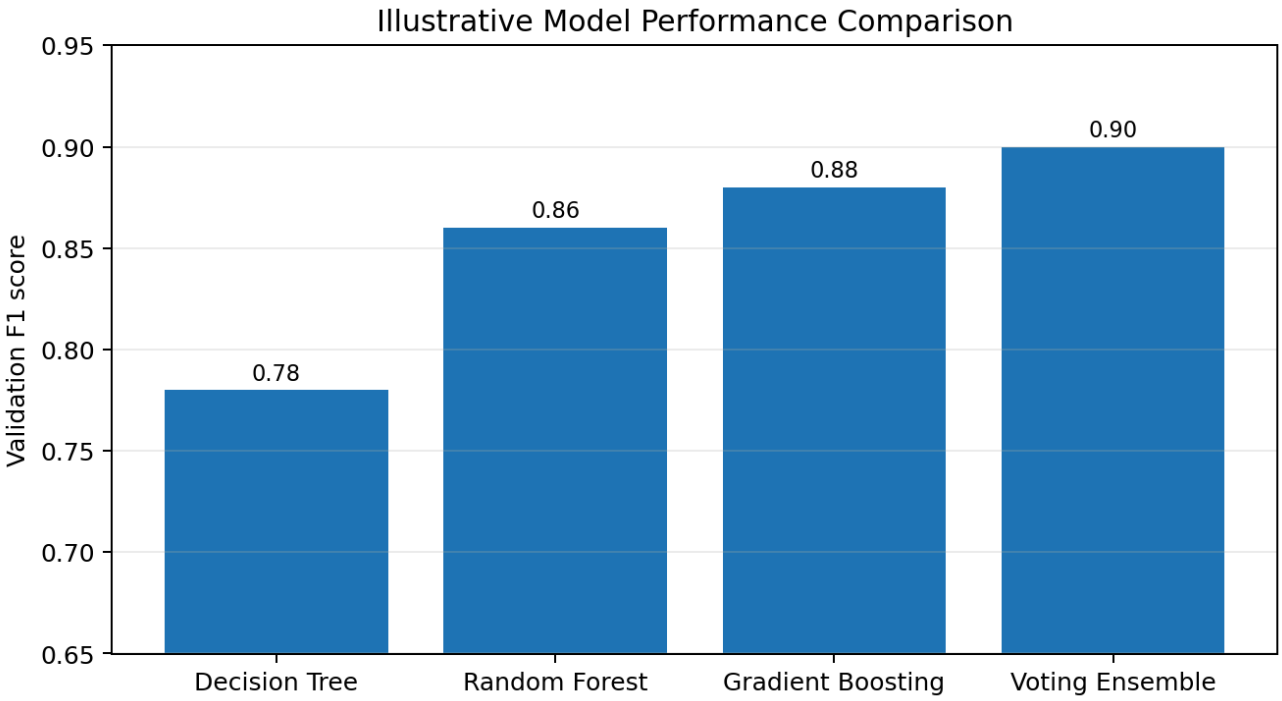


Figure 4. Illustrative comparison between a single tree and ensemble approaches.

Family	Core idea	Example	Strength
Bagging	Train many models on bootstrapped samples	Random Forest	Variance reduction and robustness
Boosting	Sequentially focus on previous errors	Gradient Boosting / AdaBoost	High predictive power on structured data
Voting	Combine predictions from diverse models	Hard or soft voting classifier	Simple way to diversify errors
Stacking	Use a meta-model on base outputs	Stacking classifier	Can learn when each base model is strongest

4. AMK Research Lab alignment

Within the AMK Research Lab model, preprocessing and model combination are not merely technical choices; they are governance choices. **Science** emphasizes data quality and validated transformations, **Intelligence** emphasizes human oversight and evidence-driven model selection, and **Innovation** emphasizes deployable pipelines that are reproducible, auditable, and safe.

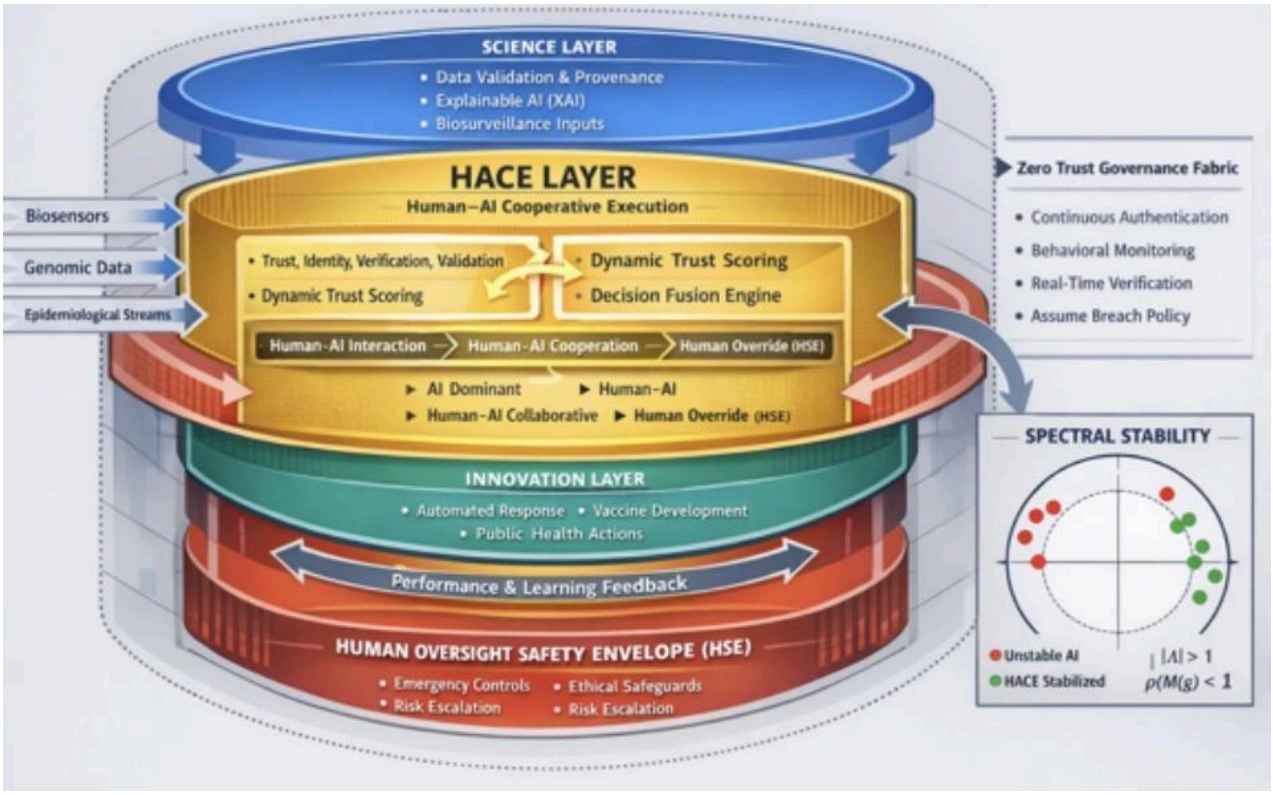


Figure 5. HACE-oriented governance view: human–AI cooperative execution and safety envelope.

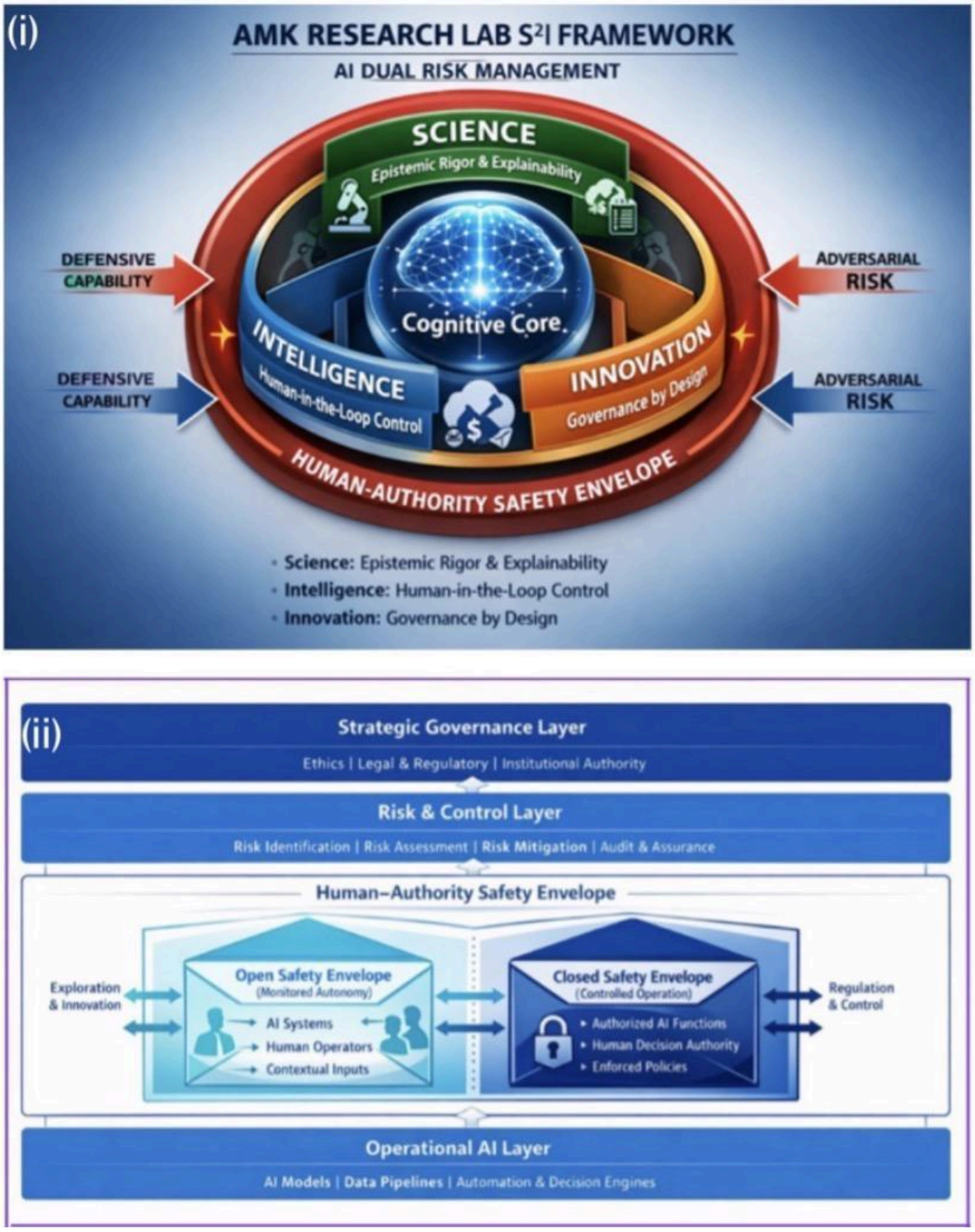


Figure 6. AMK S²I framework: Science, Intelligence, Innovation, and the human-authority safety envelope.

5. Sample Python workflow

The following example shows a simple pipeline that standardizes features, reduces dimension with PCA, and evaluates an ensemble classifier. **This code is educational and should be adapted for real datasets, validation strategy, and governance checks.**

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

X, y = load_wine(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("pca", PCA(n_components=0.95)),
    ("model", RandomForestClassifier(
        n_estimators=200, random_state=42
    )),
])

pipeline.fit(X_train, y_train)
pred = pipeline.predict(X_test)
print(classification_report(y_test, pred))
```

6. Lab activities and capstone directions

- Compare unscaled vs standardized inputs on a neural network or k-nearest-neighbors classifier and explain the difference in convergence or accuracy.
- Run PCA on a multivariate dataset, plot the cumulative explained variance, and justify the number of retained components.
- Build three models: a single tree, a random forest, and a boosting model; compare F1 score, calibration, and training time.
- Design an AMK-style report that records preprocessing decisions, PCA retention logic, ensemble choice, and responsible-deployment controls.

7. Responsible AI checkpoints

Stage	Common risk	Control
Normalization	Data leakage from fitting scalers on all data	Fit transforms only on training data and preserve pipeline artifacts
PCA	Loss of interpretability and hidden bias concentration	Document retained components and review group-wise performance
Ensembling	Over-complexity and reduced explainability	Use model cards, benchmark single-model baselines, and justify added complexity

Deployment	Drift and silent degradation	Monitor input statistics, confidence, and re-validation thresholds

References

- National Institute of Standards and Technology. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0).
- scikit-learn developers. (2026). StandardScaler documentation.
- scikit-learn developers. (2026). PCA documentation.
- scikit-learn developers. (2026). Ensemble methods documentation.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.