Ensemble Machine Learning Models: Bagging, Boosting, and Stacking

This handout provides a concise overview and Python code examples for three fundamental ensemble machine learning techniques: **Bagging**, **Boosting**, and **Stacking**. We'll use `scikit-learn` for practical implementation.

1. Setting Up: Data and Base Estimator

First, let's prepare our environment and a synthetic dataset. We'll also establish a baseline performance with a single Decision Tree.

```
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier, BaggingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import StackingClassifier
```

```
# Generate a synthetic dataset
X, y = make_classification(n_samples=1000, n_features=20, n_informative=10, n_redundant=5, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

print("Dataset shape:", X.shape)
print("Training data shape:", X_train.shape)
print("Testing data shape:", X_test.shape)

# Define a base estimator (Decision Tree) for comparison
base_estimator = DecisionTreeClassifier(max_depth=5, random_state=42)
base_estimator.fit(X_train, y_train)
y_pred_base = base_estimator.predict(X_test)
print(f"\nAccuracy of Base Estimator (Decision Tree): {accuracy_score(y_test, y_pred_base):.4f}")
```

2. Bagging (Bootstrap Aggregating)

Concept

Bagging trains multiple base models **independently** on different random subsets (with replacement, called **bootstrapping**) of the training data. The final prediction is an aggregation (e.g., averaging for regression, majority voting for classification) of the individual model predictions. This technique primarily aims to **reduce variance** and prevent overfitting.

Code Example (Random Forest)

Random Forest is a popular bagging algorithm that uses Decision Trees as base estimators and adds further randomness by considering only a subset of features at each split.

```
print("\n--- Bagging Example (Random Forest) ---")
bagging_model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42, n_jobs=-1)
bagging_model.fit(X_train, y_train)
y_pred_bagging = bagging_model.predict(X_test)
print(f"Accuracy of Bagging (Random Forest): {accuracy_score(y_test, y_pred_bagging):.4f}")

# General BaggingClassifier for other base estimators
bagging_dt_model = BaggingClassifier(
    estimator=DecisionTreeClassifier(max_depth=5, random_state=42),
    n_estimators=100,
    random_state=42,
    n_jobs=-1
)
bagging_dt_model.fit(X_train, y_train)
y_pred_bagging_dt = bagging_dt_model.predict(X_test)
print(f"Accuracy of Bagging (Decision Tree base): {accuracy_score(y_test, y_pred_bagging_dt):.4f}")
```

3. Boosting

Concept

Boosting trains models **sequentially**. Each new model focuses on correcting the errors made by the previous models. It iteratively adjusts the weights of misclassified samples or fits new models to the residuals. This technique primarily aims to **reduce bias** and convert "weak learners" into "strong learners."

Code Example (Gradient Boosting & AdaBoost)

```
print("\n--- Boosting Example ---")

# a) Gradient Boosting
# Builds an additive model by fitting new trees to the residuals (errors) of previous trees.
gradient_boosting_model = GradientBoostingClassifier(n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42)
gradient_boosting_model.fit(X_train, y_train)
y_pred_gb = gradient_boosting_model.predict(X_test)
print(f"Accuracy of Boosting (Gradient Boosting): {accuracy_score(y_test, y_pred_gb):.4f}")

# b) AdaBoost (Adaptive Boosting)
# Focuses on misclassified samples by assigning higher weights to them.
adaboost_model = AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=1), n_estimators=100, learning_rate=1.0, random_state=42)
adaboost_model.fit(X_train, y_train)
y_pred_ada = adaboost_model.predict(X_test)
print(f"Accuracy of Boosting (AdaBoost): {accuracy_score(y_test, y_pred_ada):.4f}")
```

4. Stacking

Concept

Stacking (Stacked Generalization) is a multi-level ensemble.
Level 0 (Base Models): Several **diverse** models (e.g., Logistic Regression, Decision Tree, SVM, KNN) are trained on the original training data.
Level 1 (Meta-Model/Blender): The predictions of the base models (typically out-of-fold predictions to prevent data leakage) become the input features for a new model, called the **meta-model**. The meta-model learns to combine the predictions of the base models optimally. Stacking aims to reduce both bias and variance by leveraging the strengths of diverse models.
Code Example

```
print("\n--- Stacking Example ---")

# Define diverse base models (level-0 models)
estimators = [
    ('lr', LogisticRegression(solver='liblinear', random_state=42)),
    ('dt', DecisionTreeClassifier(max_depth=5, random_state=42)),
    ('knn', KNeighborsClassifier(n_neighbors=5)),
    ('svc', SVC(probability=True, random_state=42)) # SVC needs probability=True for predict_proba
]

# Define the meta-model (level-1 model)
meta_model = LogisticRegression(solver='liblinear', random_state=42)

stacking_model = StackingClassifier(
    estimators=estimators,
    final_estimator=meta_model,
    cv=5, # Cross-validation folds for generating meta-features
    n_jobs=-1,
    passthrough=True # Meta-model also receives original features
)

stacking_model.fit(X_train, y_train)
y_pred_stacking = stacking_model.predict(X_test)
print(f"Accuracy of Stacking: {accuracy_score(y_test, y_pred_stacking):.4f}")
```

Differentiating Factors: Bagging vs. Boosting vs. Stacking

Feature/Concept	Bagging	Boosting	Stacking
Training Flow	Parallel/Independent	Sequential	Two-stage (Base models parallel, Meta-model sequential)
Error Handling	Reduces variance by averaging/majority voting	Focuses on correcting errors of previous models	Learns optimal combination to leverage strengths and reduce errors
Base Model Diversity	Usually **homogeneous** (same type)	Usually **homogeneous** (same type, often weak)	**Heterogeneous** (diverse types)

Feature/Concept	Bagging	Boosting	Stacking
Combination Method	Simple aggregation (averaging/voting)	Weighted averaging (weights learned iteratively)	A separate **meta-model** learns the optimal combination
Primary Goal	Reduce **Variance**	Reduce **Bias**	Improve overall predictive power by combining diverse perspectives
Example	Random Forest	Gradient Boosting, AdaBoost, XGBoost, LightGBM	Stacking Classifier/Regressor

This handout covers the core concepts and practical implementations of Bagging, Boosting, and Stacking. Choosing the right ensemble method depends on your specific problem, data characteristics, and desired trade-off between bias and variance.