Lesson 15: Ensemble Methods - Boosting for Regression

Topics Covered:

1️⃣ **Understanding Ensemble Methods: Bagging vs. Boosting**
2️⃣ **Gradient Boosting for Regression (GBR)**
3️⃣ **AdaBoost for Regression**
4️⃣ **Hyperparameter Tuning for Boosting Algorithms**
5 **Performance Evaluation with MSE, RMSE, MAE, R^2**

1️⃣ Bagging vs. Boosting: What's the Difference?

Bagging (Bootstrap Aggregating) and **Boosting** are both **ensemble learning techniques**, but they work differently:

Feature	Bagging	Boosting
Purpose	Reduce variance (overfitting)	Reduce bias (underfitting)
Model Dependency	Models trained **independently**	Models trained **sequentially**
Weight Adjustment	Equal weight for all models	Adjusts weights based on errors
Example Algorithms	Random Forest	AdaBoost, Gradient Boosting, XGBoost

🔹 **Bagging reduces variance** by averaging multiple models.

🔹 **Boosting reduces bias** by training weak models sequentially, each improving upon the previous one.

2️⃣ Gradient Boosting for Regression (GBR)

Gradient Boosting is an advanced boosting technique where:
✅ Each new model **corrects the errors** of the previous model.
✅ Uses **gradient descent** to minimize the loss function.
✅ Works well with **small and medium datasets**.

📌 Steps in Gradient Boosting

1. Start with a weak model (usually a Decision Tree 🌳).
2. Compute the **error residuals** (difference between actual and predicted values).
3. Train a new model to predict these **residuals**.

4. Update the final prediction by adding the new model's prediction.
5. Repeat the process for multiple iterations.

3□ AdaBoost for Regression

AdaBoost (**Adaptive Boosting**) is another boosting method where:
✅ It **assigns higher weights** to misclassified points.
✅ Each model is trained sequentially, **focusing more on difficult samples**.
✅ Works well when **data has noise or outliers**.

4□ Implementing Gradient Boosting & AdaBoost for Regression

Let's use **Gradient Boosting** and **AdaBoost** on a **car price dataset**.

📌 Step 1: Import Libraries

```
import pandas as pd
import numpy as np
import time
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor, AdaBoostRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load dataset
df = pd.read_csv('car_price.csv')  # Replace with actual dataset

# Select features and target variable
X = df[['mileage', 'year', 'engine_size']]  # Example features
y = df['price']  # Target variable

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, 
random_state=42)
```

📌 Step 2: Train Gradient Boosting Model

```
# Initialize Gradient Boosting model
gbr = GradientBoostingRegressor(n_estimators=500, learning_rate=0.05, 
max_depth=4, random_state=42)

# Train the model
gbr.fit(X_train, y_train)

# Predictions
y_train_pred_gbr = gbr.predict(X_train)
```

```
y_test_pred_gbr = gbr.predict(X_test)
```

📌 Step 3: Train AdaBoost Model

```
# Initialize AdaBoost model
ada = AdaBoostRegressor(n_estimators=500, learning_rate=0.05, random_state=42)

# Train the model
ada.fit(X_train, y_train)

# Predictions
y_train_pred_ada = ada.predict(X_train)
y_test_pred_ada = ada.predict(X_test)
```

📌 Step 4: Evaluate Performance (MSE, RMSE, MAE, R^2)

```
def evaluate_model(y_true, y_pred, model_name):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)

    print(f"{model_name} Performance:")
    print(f"  MSE: {mse:.2f}")
    print(f"  RMSE: {rmse:.2f}")
    print(f"  MAE: {mae:.2f}")
    print(f"  R²: {r2:.2f}")
    print("-" * 40)

# Evaluate Gradient Boosting
evaluate_model(y_test, y_test_pred_gbr, "Gradient Boosting")

# Evaluate AdaBoost
evaluate_model(y_test, y_test_pred_ada, "AdaBoost")
```

5□ Hyperparameter Optimization (Grid Search)

To further improve performance, we can **tune hyperparameters**.

```
# Define hyperparameters to search for Gradient Boosting
param_grid = {
    'n_estimators': [100, 300, 500],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 5]
}

grid_search = GridSearchCV(GradientBoostingRegressor(), param_grid, cv=3,
scoring='r2', n_jobs=-1)
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
```

```
# Train model with best parameters
best_gbr = grid_search.best_estimator_
y_test_pred_best_gbr = best_gbr.predict(X_test)

# Evaluate Optimized Model
evaluate_model(y_test, y_test_pred_best_gbr, "Optimized Gradient Boosting")
```

📌 Key Takeaways

✓ **Bagging vs. Boosting**: Bagging reduces variance; Boosting reduces bias.
✓ **Gradient Boosting**: Uses gradient descent to minimize errors.
✓ **AdaBoost**: Focuses on difficult data points by re-weighting them.
✓ **Hyperparameter Tuning**: Improves model performance by finding the best parameters.
✓ **Evaluation Metrics**: Always use **MSE, RMSE, MAE, R^2** to assess model accuracy.