Predicting EV Battery Life Using `GradientBoostingRegressor`

What is Gradient Boosting?

- Combines many small decision trees to make better predictions.
- Each tree fixes the errors of the last one.
- Controlled by learning rate, tree depth, and number of trees.

Step 1: Load EV Data

We use temperature and battery age to predict battery life.

```
import pandas as pd

data = pd.DataFrame({
    'Temperature': [25, 30, 20, 35, 22],
    'Battery_Age': [2, 3, 1, 4, 2],
    'Battery_Life': [5, 4, 6, 3, 5]
})
```

Step 2: Define X and y

Split features and target.

```
X = data[['Temperature', 'Battery_Age']]
y = data['Battery_Life']
```

Step 3: Train/Test Split

Split data to evaluate the model later.

```
from sklearn.model_selection import
    train_test_split

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

Step 4: Train the Model

Create a Gradient Boosting model and fit it to the training data.

```
from sklearn.ensemble import
    GradientBoostingRegressor

model = GradientBoostingRegressor(
     learning_rate=0.1,
     n_estimators=100,
     max_depth=2,
     random_state=42
)
model.fit(X_train, y_train)
```

Step 5: Predict and Evaluate

Measure how well our model predicts battery life.

```python
from sklearn.metrics import mean_squared_error

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("MSE:", mse)
```

Step 5 Results: Evaluation Output

Mean Squared Error:

$$\text{MSE} = 0.999986719344651$$

Step 6: Compare Predictions

```
results = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted': y_pred
})
print(results)
```

Step 6 Results: Actual vs Predicted

Prediction Output Table:

Actual	**Predicted**
4	5.0

Note: The predicted value is close but slightly off — this is expected due to limited data.

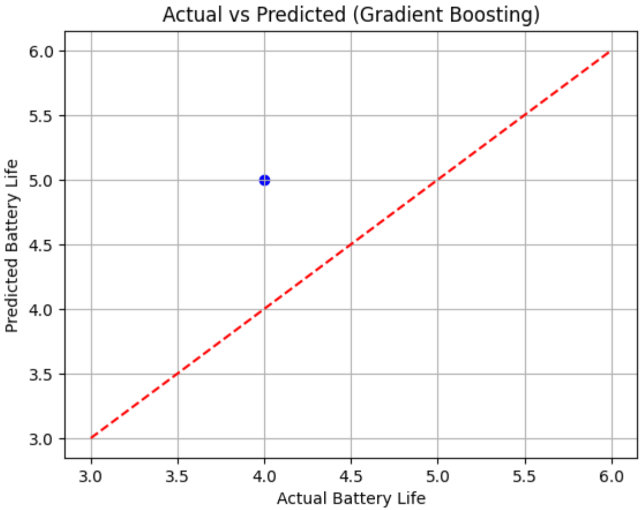
Step 7: Visualize Predictions

Plot actual vs predicted battery life.

```
import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred)
plt.plot([min(y), max(y)], [min(y), max(y)], 'r
    --')
plt.xlabel("Actual Battery Life")
plt.ylabel("Predicted")
plt.title("Actual vs Predicted")
plt.grid(True)
plt.show()
```

Step 7 Result: Visualization



Red dashed line = perfect prediction.Blue points = model predictions.

Summary

- Gradient Boosting builds multiple trees step by step.
- Each new tree corrects previous prediction errors.
- Scikit-learn makes this easy using `GradientBoostingRegressor`.
- Visualizing predictions helps validate the model.