Tutorial: Estimating EV Battery Life Using Gradient Boosting

May 28, 2025

Overview

This tutorial demonstrates how to use Gradient Boosting Regressor to estimate the remaining useful life (RUL) of EV batteries. We use synthetic data to simulate battery health metrics and train a regression model.

1. Setup and Imports

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

2. Synthetic Data Generation

```
np.random.seed(42)
...
df.head()
```

3. Exploratory Data Analysis (EDA)

```
sns.pairplot(df, vars=['cycle_count', 'avg_temp', 'internal_resistance', 'remaining_life'])
plt.show()

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```

Visualization: Pairplot

This plot shows pairwise relationships between important features and the target variable `remaining_life`.

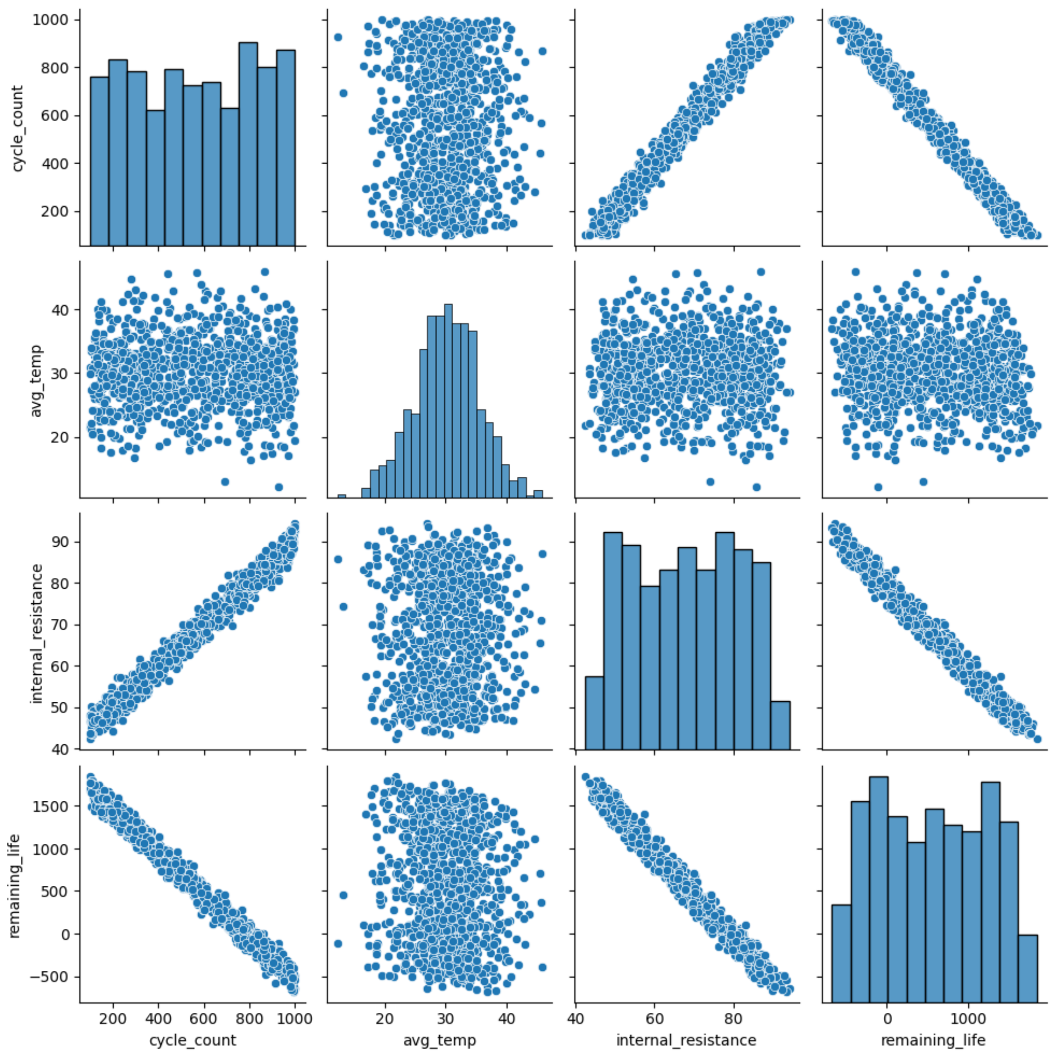
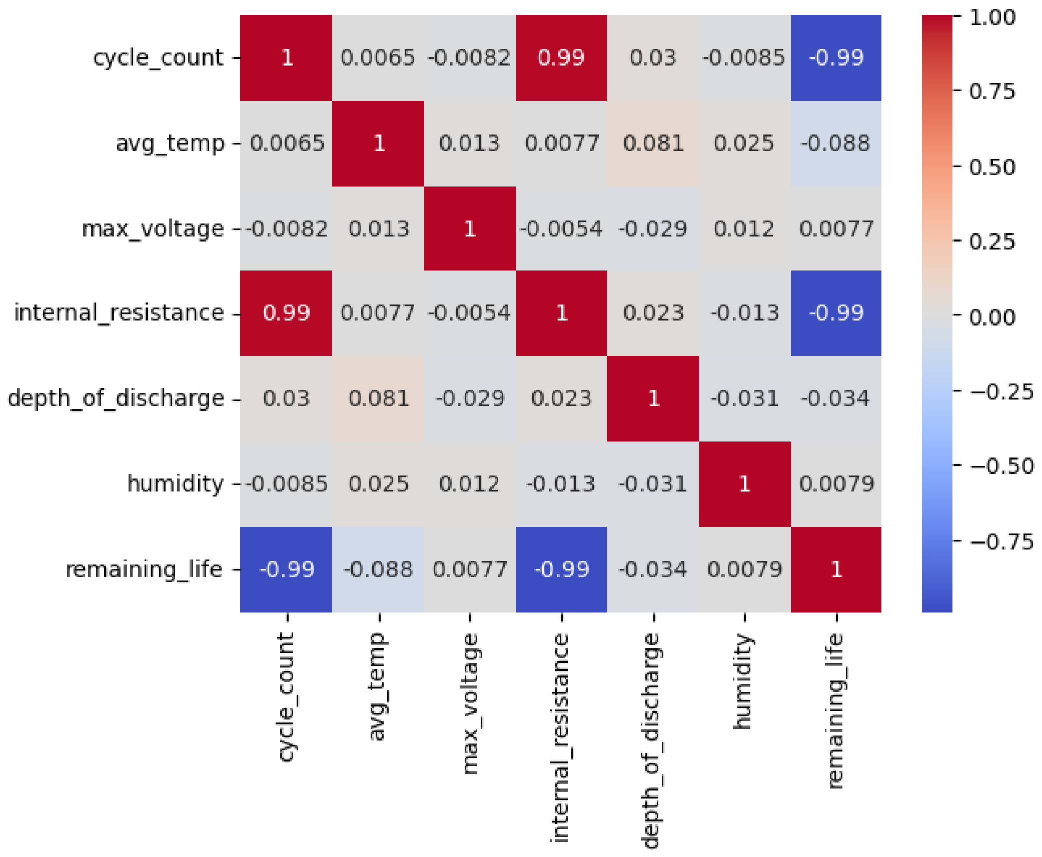


Figure 1: Pairplot of Key Features and Target

Visualization: Heatmap

This heatmap shows correlation between numerical features. Strong linear relationships are easy to detect.



	cycle_count	avg_temp	max_voltage	internal_resistance	depth_of_discharge	humidity	remaining_life
cycle_count	1	0.0065	-0.0082	0.99	0.03	-0.0085	-0.99
avg_temp	0.0065	1	0.013	0.0077	0.081	0.025	-0.088
max_voltage	-0.0082	0.013	1	-0.0054	-0.029	0.012	0.0077
internal_resistance	0.99	0.0077	-0.0054	1	0.023	-0.013	-0.99
depth_of_discharge	0.03	0.081	-0.029	0.023	1	-0.031	-0.034
humidity	-0.0085	0.025	0.012	-0.013	-0.031	1	0.0079
remaining_life	-0.99	-0.088	0.0077	-0.99	-0.034	0.0079	1

Figure 2: Feature Correlation Heatmap

4. Train-Test Split

```
X = df.drop(columns=['remaining_life'])
y = df['remaining_life']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. Training Gradient Boosting Regressor

```
gbr = GradientBoostingRegressor(n_estimators=200, learning_rate=0.1, max_depth=4,
    random_state=42)
gbr.fit(X_train, y_train)
```

6. Evaluation

```
y_pred = gbr.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
...
```

Visualization: Predicted vs True Remaining Life

The scatter plot compares predicted values to actual values. Points near the red dashed line represent better predictions.

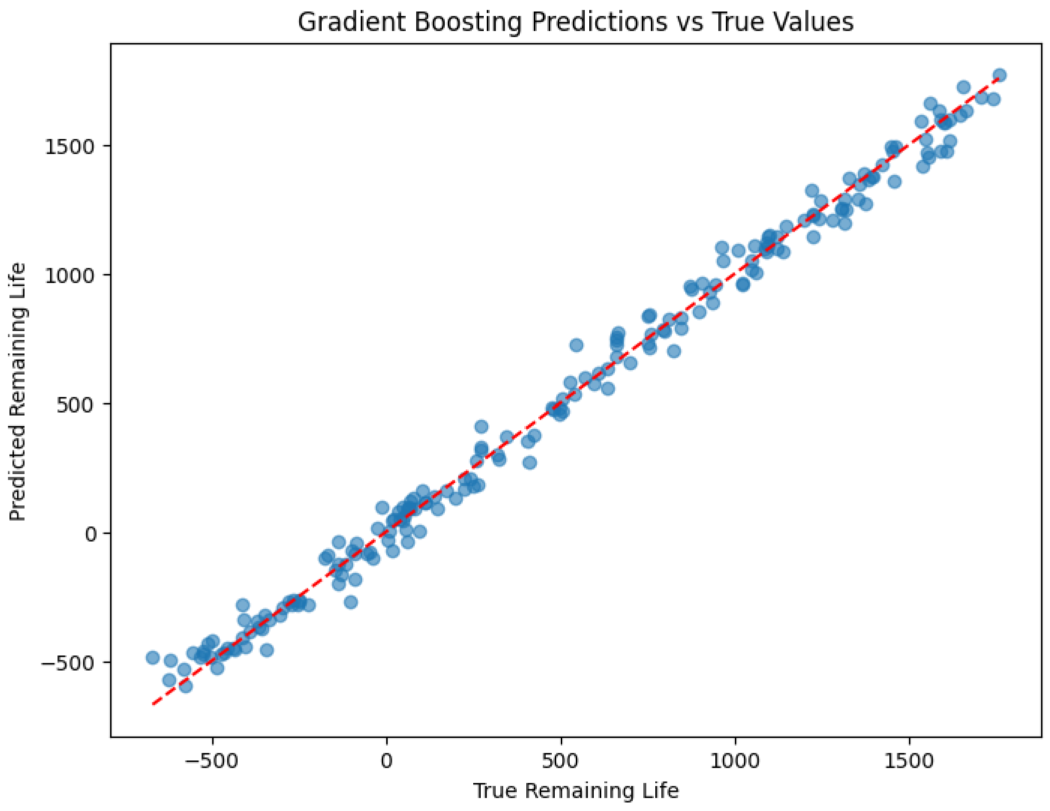


Figure 3: Predicted vs True Remaining Life

7. Residual Analysis

```
residuals = y_test - y_pred
sns.histplot(residuals, kde=True)
plt.title("Residual Distribution")
plt.show()
```

Visualization: Residual Distribution

A roughly symmetric and narrow distribution indicates a good model fit.

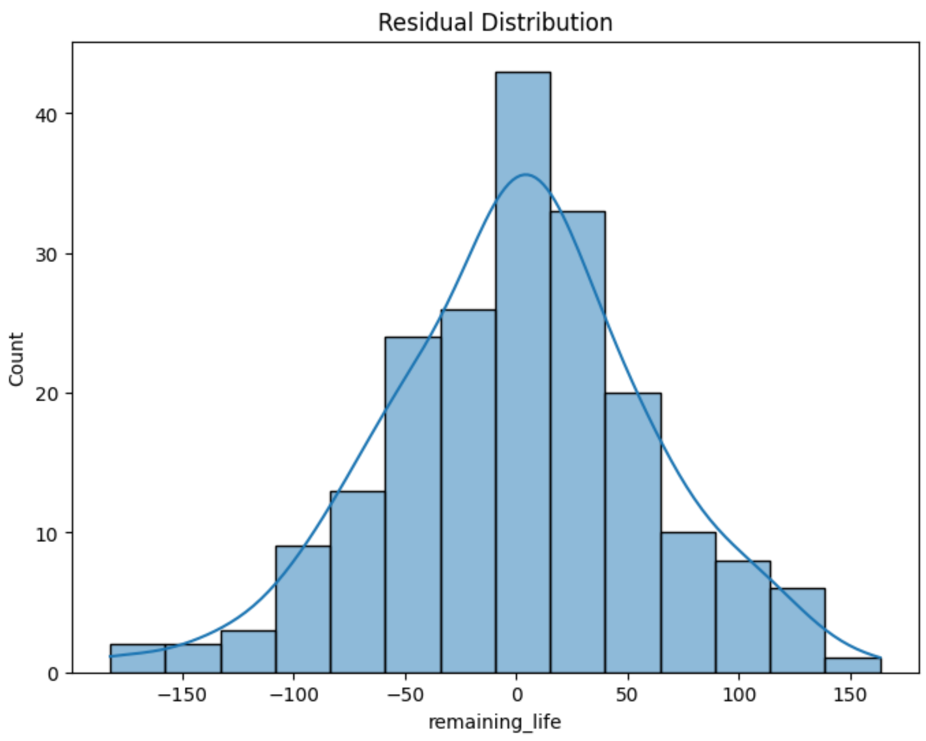


Figure 4: Distribution of Residuals

8. Save Model for Deployment

```
import joblib
joblib.dump(gbr, 'ev_battery_life_gbr_model.pkl')
```