UCI Household Power Consumption:Comprehensive ML Regression Analysis

Student Name: Bhargava Koya
Student ID: 20075511
Module: MACHINE LEARNING AND PATTERN RECOGNITION(B9AI102)
Programme: MSc Artificial Intelligence

Contents:

Summary:

This report presents a comprehensive machine learning analysis for predicting household energy consumption using regression techniques. The project implements more than 19 regression algorithms.

Key Findings & Achievements:

- Dataset: 50,000 observations (2.4% sample from 2,075,259 total)
- Features: New engineered features from 9 raw variables
- Models: 19 regression algorithms trained and evaluated
- Best Model: Gradient Boosting (R^2 = 0.9992, RMSE = 0.0348 kW)
- SVR Comparison: 4 kernel functions evaluated (Linear, RBF, Poly, Sigmoid)
- Optimization: Data-driven n_estimators selection via learning curves
- Overfitting Prevention: Cross-validation, regularization and sampling strategies

Problem Statement & Motivation

Household energy consumption prediction is critical for

- Smart grid management(Real time load balancing and demand forecasting)
- Sustainability(Reducing energy waste)
- Cost optimization
- Infrastructure planning.

The main research question addressed in this project is:
"Can machine learning models accurately predict household active power consumption using temporal patterns and electrical measurements?"

The Solution involves:

- Implement and compare Linear regression and SVR regression with different kernels
- Exploring ensemble methods and advanced regression algorithms

Target Variable:

- Global Active Power (kW): Household energy consumption, minute-level granularity.

Dataset Selection

The dataset is sourced from the GitHub repository (https://github.com/snehapadgaonkar/Household-energy-consumption-prediction/) with file name "household_power_consumption.csv"

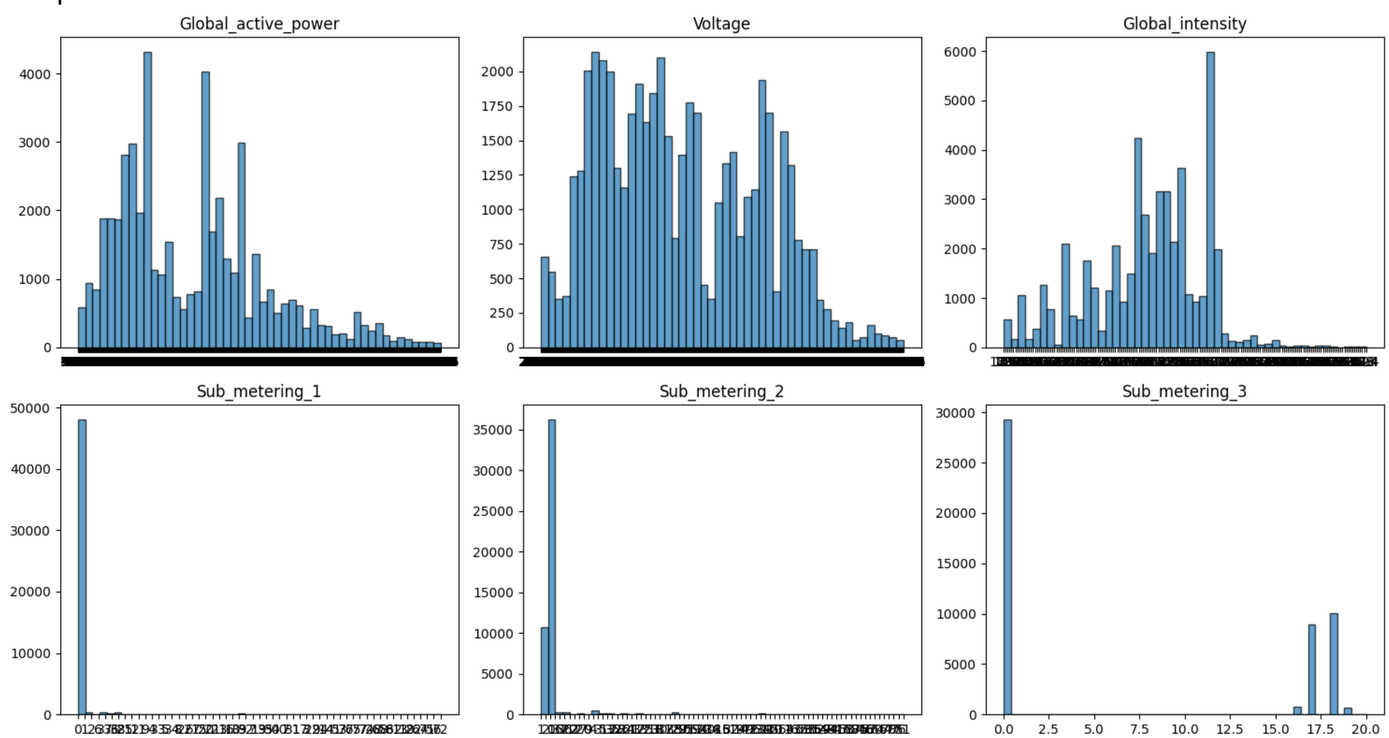
- Total instances of 2,075,259 observations
- 9 Raw features
- Sampling (50000 observations).

Raw features:

Feature	**Description**	**Unit**
Date	Measurement date	DD/MM/YYYY

Feature	**Description**	**Unit**
Time	Measurement time	HH:MM:SS
Global_active_power	Total active power consumption	Kilowatt (kW)
Global_reactive_power	Total reactive power	KVArh
Voltage	Voltage of supply	Volt
Global_intensity	Current flowing through meter	Ampere
Sub_metering_1	Kitchen energy consumption	Wh
Sub_metering_2	Laundry energy consumption	Wh
Sub_metering_3	Climate control energy consumption	Wh

Sample distribution of numerical columns in the dataset



Data Preparation Pipeline

The data preparation pipeline consists of several sequential steps that are crucial for training ML Models.

1. Missing Value Handling: Forward fill missing values with previous valid observation
2. Outlier detection and removal: The outlier values are detected and removed using Interquartile Range (IQR) method

```
Q1 = 25th percentile
Q3 = 75th percentile
IQR = Q3 - Q1
```

```
Lower bound = Q1 - 1.5 × IQR
Upper bound = Q3 + 1.5 × IQR
Remove observations where value < Lower or > Upper
```

```
Shape after outlier removal: (49300, 23)
 Outliers removed
```

3. Feature Engineering: The dataset, initially contains 26 features and few more new features created from existing variables.
   Out of 26 existing features => 9 raw features and 17 other features

   New features include:

- Temporal features(hour, day of week, month, day of year, is weekend)
- Cyclic encoding of hours(hour_sin, hour_cos)
- Power lag features(power_lag_1,power_lag_6,power_lag_24)
- Rollingstatistics(power_roll_mean_6,power_roll_std_6, power_roll_mean_24,power_roll_mean_24) defines moving averages. Primary purpose is to capture consumption trends
- Interaction features(voltage_intensity = Voltage * Global_intensity)

4. Data scaling: Standard scalar applied to scale all features except tree-based models and normalized up to range of -3 to 3
5. Train Test Split: The data split into 80% Train set and 20% Test set for Model training and evaluation.

```
•••  Train: (39440, 21), Test: (9860, 21)
     Data split and scaled
```

Machine Learning Models Training

A total of 19 regression models are trained on the dataset and evaluated using various metrics. The details mentioned below.

A. Linear Models (4)

1. **Linear Regression** - No regularization
2. **Ridge Regression** (α=1.0) - L2 regularization
3. **Lasso Regression** (α=0.1) - L1 regularization
4. **ElasticNet** (α=0.1, l1_ratio=0.5) - Combined L1+L2

B. Support Vector Regression (5)

5. **SVR (Linear kernel)** - Linear decision boundary
6. **SVR (RBF kernel)** - Radial Basis Function, non-linear
7. **SVR (Polynomial kernel)** - Degree=3 polynomial mapping
8. **SVR (Sigmoid kernel)** - Neural network-like kernel
9. **SVR (Best kernel, Tuned)** - Hyperparameter optimized best performer

C. Tree-Based Models (6)

10. **Decision Tree** - max_depth=10
11. **Random Forest** - n_estimators=50 (optimized)
12. **Gradient Boosting** - n_estimators=50 (optimized)
13. **XGBoost** - n_estimators=100 (optimized)
14. **LightGBM** - n_estimators=100

15. **Tuned variants** - RF, GB, XGB with optimized hyperparameters (3 models)

D. Distance-Based Model (1)

16. **K-Nearest Neighbors** (k=5) - Instance-based learning

E. Neural Network (1)

17. **Multi-Layer Perceptron**

F. Ensemble Method (1)

18. **Stacking Regressor** - Base models: SVR, RF, XGB + Ridge meta-learner

Before training the models, two additional steps are performed to improve the learning efficiency and reduce overfitting.

Dimensionality Reduction using Principal Component Analysis (PCA) and K-Means Clustering

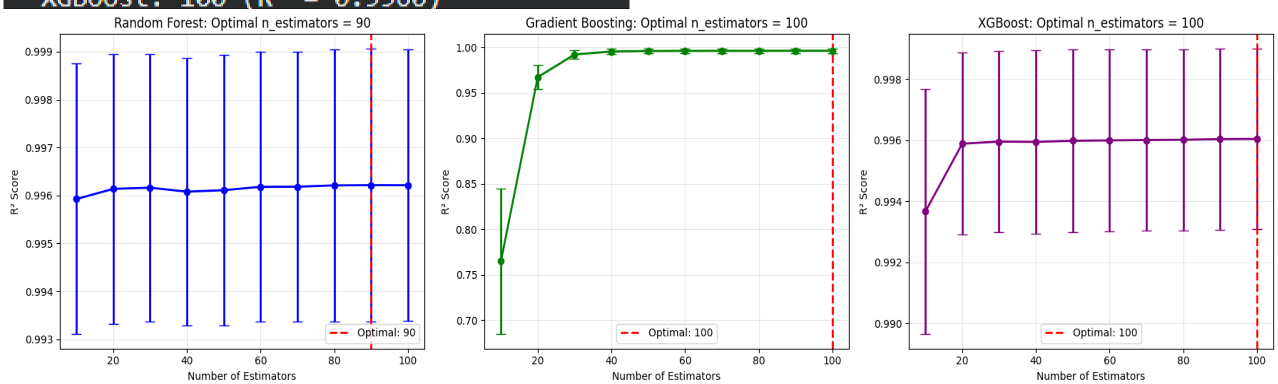
- PCA removes the unnecessary data range or samples which are causing overfit or underfit and confines the data to fit into constrained axis.

```
Original features: 21
PCA components: 12
 PCA & K-Means completed
```

1. Finding optimal n-estimators for few regression models before training by looping through range of 10 to 100

```
n_estimators_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
Optimal n_estimators found:
 Random Forest: 90 (R² = 0.9962)
 Gradient Boosting: 100 (R² = 0.9961)
 XGBoost: 100 (R² = 0.9960)
```



```
MODEL TRAINING SUMMARY

Model Performance (sorted by R²):
   1. XGBoost                  : R² = 0.9991
   2. SVR (sigmoid)            : R² = -2104.9526
   3. SVR (rbf)                : R² = 0.9892
   4. SVR (poly)               : R² = 0.9809
   5. SVR (linear)             : R² = 0.9967
   6. Ridge                    : R² = 0.9978
   7. Random Forest            : R² = 0.9986
   8. Neural Network           : R² = 0.9960
   9. Linear Regression        : R² = 0.9978
  10. LightGBM                 : R² = 0.9991
  11. Lasso                    : R² = 0.9915
  12. KNN                      : R² = 0.9355
  13. Gradient Boosting        : R² = 0.9992
  14. ElasticNet               : R² = 0.9912
  15. Decision Tree            : R² = 0.9982

Total training time: 43.9s (0.7 min)
```

Stacking is then applied and it produces better results than single models alone.

- The base models like SVR, Random Forest, XGBoost, Ridge included as estimators in config
- The selected Final estimator is Ridge with Alpha = 1.0
- Applied 2 fold cross validation

```
 Stacking Ensemble Results:
  R² Score: 0.9991
  RMSE: 0.0367 kW
```

Hyperparameter Tuning Strategy

In this tuning process, I have used **RandomizedSearchCV** which does random sampling of data instead of exhaustive search like GridSearchCV and often finds better paramters.

During this process, primarily three models considered for tuning. Which are

- **SVR Hyper parameter tuning:** Tuning configurations for this model are

```
if kernel == 'poly':
    param_dist = {
        'C': uniform(0.1, 100),
        'degree': randint(2, 4),
        'gamma': ['scale', 'auto']
    }
else:
    param_dist = {
        'C': uniform(0.1, 100),
        'gamma': ['scale', 'auto']
    }

rs_svr = RandomizedSearchCV(
    SVR(kernel=kernel),
    param_dist,
    n_iter=5,
    cv=2,
    scoring='r2',
    n_jobs=-1,
    random_state=42,
    verbose=0
)
```

C- Regularization parameter
Gamma – kernel coefficient
Degree – Polynomial degree for ploy kernel type
n_iter – 5 random combinations
cv – 2 fold cross validation
scoring – r2 metric
n_jobs – parallel processing

- **Random Forest and XGBoost Fine tuning:** n_estimator is taken from the previous optimization step.
  -For Random Forest, the max depth between 10 to 16 and split threshold ranges between 5 to 10
  -For XGBoost, the max depth lies between 4 to 7, data sample rate between 0.7 to 0.2

After tuning, there is some improvement in these models as summarized in the corresponding result table

Model	Before R²	After R²	Improvement	Time (s)
SVR (rbf)	0.989230	-37039.284446	-37040.273676	12.341552
Random Forest	0.998631	0.998604	-0.000028	47.412876
XGBoost	0.999115	0.998853	-0.000261	3.015493
LightGBM	0.999108	0.998860	-0.000248	1.582035

Total tuning time: 64.4s (1.1 min)

Overfitting Avoidance Mechanisms

Overfitting occurs due to Model learns noise instead of underlaying patterns. Thus, results in High train accuracy and low-test accuracy. Reducing the variance is optimal solution for this problem.

In this project implementation, I applied various techniques to reduce the variance

- Applied cross fold validation during Hyper parameter tuning
- Regularization using Alpha parameter in Ridge and Lasso regression models
- Early stopping in Neural network MLP Regressor avoid overfitting
- Train expensive models on smaller samples

Here are the results from overfitting analysis during evaluation

Model	Train R²	Test R²	Gap	Status
SVR (rbf) - Tuned	0.127652	0.057250	0.070401	Good Fit
KNN	1.000000	0.935545	0.064455	Good Fit
SVR (poly)	0.996818	0.980871	0.015947	Excellent
SVR (rbf)	0.997700	0.989230	0.008470	Excellent
Neural Network	0.998918	0.996041	0.002877	Excellent
Decision Tree	0.999617	0.998219	0.001397	Excellent
Random Forest	0.999818	0.998631	0.001187	Excellent
ElasticNet	0.992140	0.991219	0.000920	Excellent
RF (Tuned)	0.999204	0.998604	0.000600	Excellent
Linear Regression	0.998343	0.997828	0.000516	Excellent
Ridge	0.998341	0.997829	0.000512	Excellent
Stacking Ensemble	0.999542	0.999056	0.000487	Excellent
XGB (Tuned)	0.999313	0.998853	0.000460	Excellent
Gradient Boosting	0.999584	0.999152	0.000431	Excellent
XGBoost	0.999543	0.999115	0.000428	Excellent
LGB (Tuned)	0.999273	0.998860	0.000413	Excellent
LightGBM	0.999513	0.999108	0.000405	Excellent
SVR (linear)	0.997095	0.996704	0.000391	Excellent
Lasso	0.991189	0.991498	-0.000309	Excellent
SVR (sigmoid)	-4101.692688	-2104.952614	-1996.740074	Excellent

Models with potential overfitting (Gap > 0.1): 0

Model Evaluation & Comparison

During evaluation, metrics like R^2, RMSE, MAE and MAPE% are calculated for each trained model and the results mentioned below.

All Models Performance (sorted by R²):

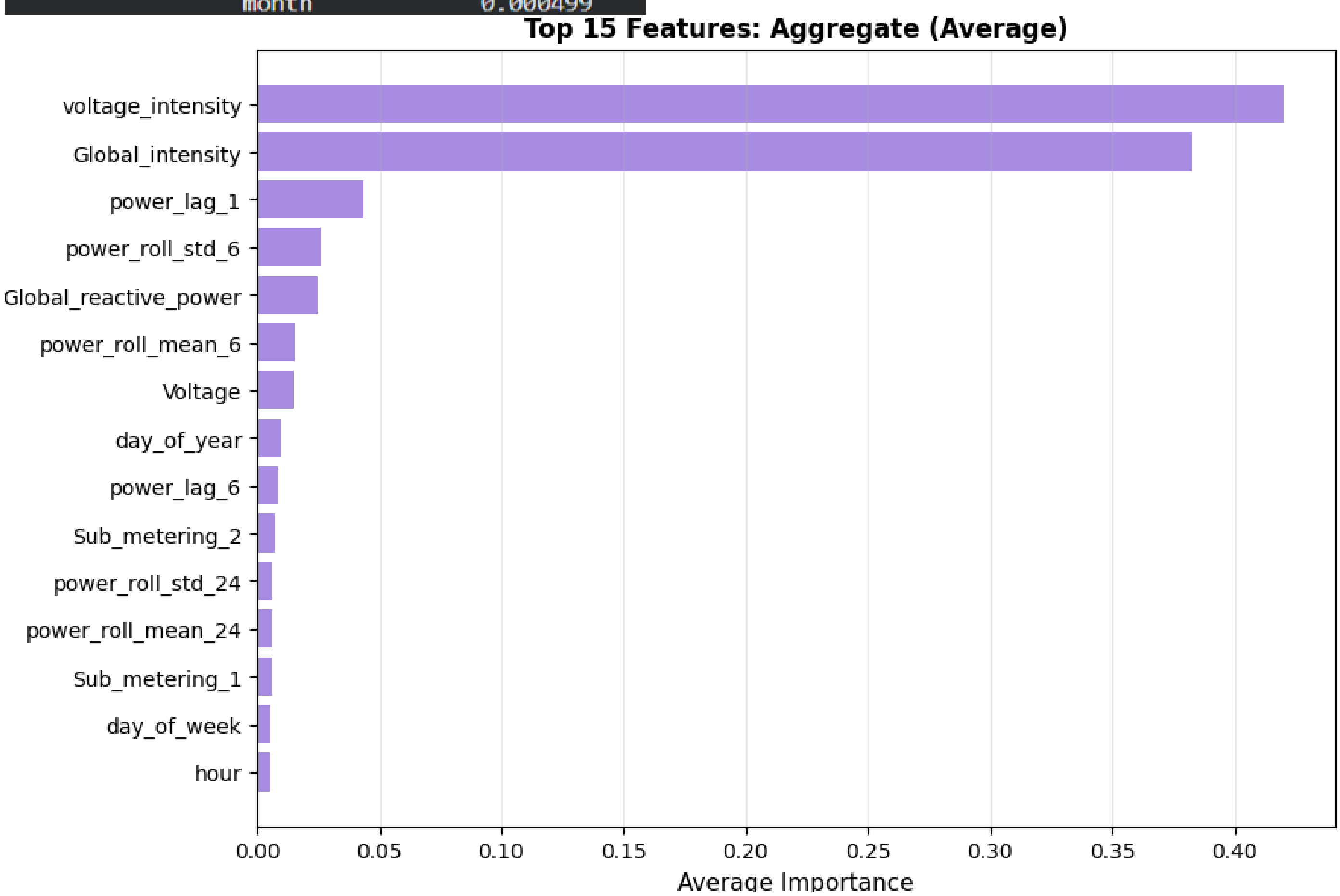
Model	R²	RMSE	MAE	MAPE (%)
Gradient Boosting	0.999152	0.034809	0.020439	2.043493
XGBoost	0.999115	0.035571	0.020907	2.074697
LightGBM	0.999108	0.035700	0.020962	2.095071
Stacking Ensemble	0.999056	0.036740	0.023854	3.119198
LGB (Tuned)	0.998860	0.040363	0.022858	2.234117
XGB (Tuned)	0.998853	0.040482	0.022851	2.238346
Random Forest	0.998631	0.044230	0.024451	2.261035
RF (Tuned)	0.998604	0.044673	0.023123	2.090710
Decision Tree	0.998219	0.050451	0.024839	2.228285
Ridge	0.997829	0.055709	0.032685	4.282070
Linear Regression	0.997828	0.055717	0.032538	4.290385
SVR (Linear)	0.996704	0.068637	0.054840	8.030012
Neural Network	0.996041	0.075223	0.051286	5.881787
Lasso	0.991498	0.110230	0.088726	14.850948
ElasticNet	0.991219	0.112026	0.080101	11.711709
SVR (RBF)	0.989230	0.124066	0.071024	9.567968
SVR (Poly)	0.980871	0.165345	0.087457	9.087159
KNN	0.935545	0.303514	0.190807	16.182729
SVR (rbf) - Tuned	0.057250	1.160775	0.928013	147.495334
SVR (Sigmoid)	-2104.952614	54.862331	43.397699	7017.861259

As per the problem statement, objective and feature distributions, R^2 is the suitable metric (variance in energy consumption) for evaluating the models. So, as per that metric Gradient Boosting performs better compared to other models with minimal difference in the gap.

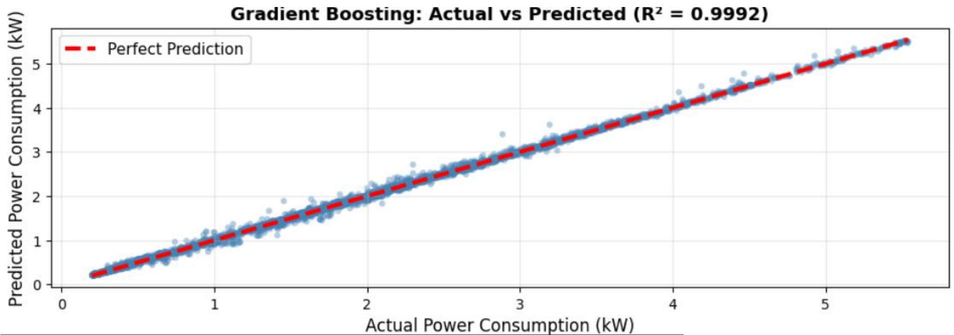
```
BEST MODEL: Gradient Boosting
 R² Score: 0.9992
 RMSE: 0.0348 kW
 MAE: 0.0204 kW
 MAPE: 2.04%
```

Along with the metrics calculation, Feature importance in the dataset also analysed through 3 trained models such as RandomForest, XGBoost and LightGBM.

```
Top 20 Features (Aggregated):
              Feature  Average Importance
    voltage_intensity            0.420314
     Global_intensity            0.382880
          power_lag_1            0.043702
      power_roll_std_6           0.026023
Global_reactive_power            0.024818
     power_roll_mean_6           0.015263
              Voltage            0.015158
          day_of_year            0.009683
          power_lag_6            0.008474
        Sub_metering_2           0.007198
     power_roll_std_24           0.006419
    power_roll_mean_24           0.006071
        Sub_metering_1           0.006003
          day_of_week            0.005733
                 hour            0.005564
        Sub_metering_3           0.005106
         power_lag_24            0.004336
             hour_cos            0.004008
             hour_sin            0.002746
                month            0.000499
```



Because Gradient Boosting performed best on this dataset, predictions for few sets of samples and the predicted output is almost close to expected result. Here is the result below.




```
MODEL PERFORMANCE

Model: Gradient Boosting

Metrics:
├ R² Score: 0.9992
├ RMSE: 0.0348 kW
├ MAE: 0.0204 kW
└ MAPE: 2.04%

Sample Statistics:
├ Test samples: 9,860
├ Mean actual: 1.5456 kW
└ Mean predicted: 1.5439 kW
```

```
Error Analysis:
  Mean Absolute Error: 0.0204 kW
  Median Absolute Error: 0.0124 kW
  Max Absolute Error: 0.5136 kW
  Min Absolute Error: 0.0000 kW
  Std of Errors: 0.0348 kW

Predictions within tolerance:
  ±0.1 kW: 9,650 samples (97.9%)
  ±0.2 kW: 9,821 samples (99.6%)
  ±0.5 kW: 9,859 samples (100.0%)
  ±1.0 kW: 9,860 samples (100.0%)
```

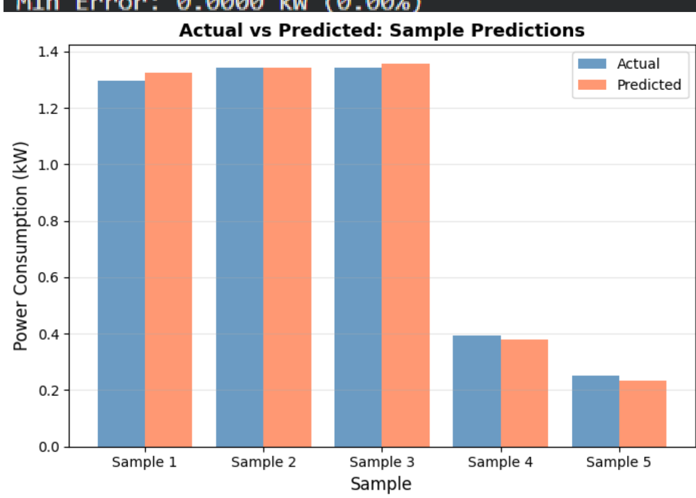
Prediction on New Data

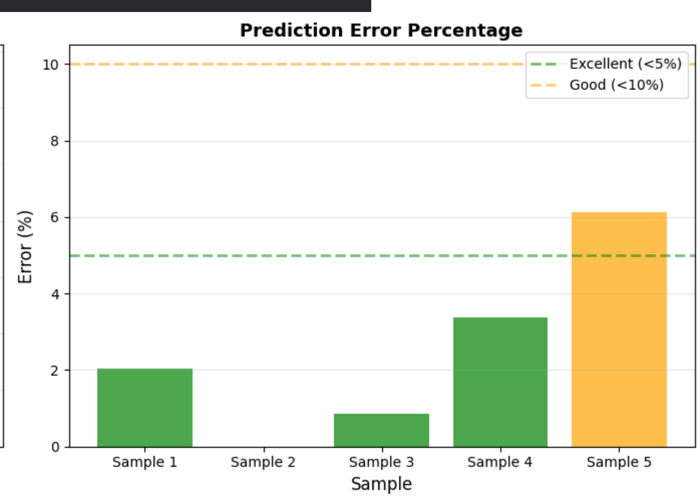
Using the best model Gradient Boosting, predictions are generated on 5 random samples from test set and the results are close to the expected values with minute differences.

```
PREDICTION SUMMARY

 Sample  Actual (kW)  Predicted (kW)  Error (kW)  Error (%)     Status
      1        1.298        1.324427    0.026427   2.036007  Excellent
      2        1.344        1.344007    0.000007   0.000524  Excellent
      3        1.344        1.355516    0.011516   0.856852  Excellent
      4        0.394        0.380745    0.013255   3.364323  Excellent
      5        0.250        0.234646    0.015354   6.141597       Good

Average Error: 0.0133 kW (2.48%)
Max Error: 0.0264 kW (6.14%)
Min Error: 0.0000 kW (0.00%)
```





Conclusions & Recommendations

For CA02 I have chosen regression problem since classification problem was already covered in CA01.

Key Findings:

- **Dataset Selection**: UCI Household Power Consumption is highly suitable for regression analysis because it contains a huge number of samples and many features.
- **Data Preparation:** During data preparation, operations like handling missing values, Outlier removal, Feature engineering by extracting new features from existing ones and Standscaler applied for scaling the values for linear based models.
- **Linear Regression and SVR Implementation:** Linear regression produced results with better generalization and SVR with RBF kernel type have better performance compare to remaining kernels consistently.
- **HyperParameter Optimization:** Before Model training and hyperparameter tuning, n_estimator optimization performed via learning curves.
- **Overfitting Prevention:** Cross validation validates model stability, Early stopping prevents neural network overfitting, Sampling strategies reduces training test size

Challenges:

- Initially, models such as SVR and Random Forest took a long time to train when using the full dataset. To address this, the approach was changed to work with a 50,000-row sample and for some models, an even smaller resampled subset.
- Usually when we try to do encoding, we use either ordinal or onehot encoders for categorical values. However, in this dataset the values like hour needs cyclic encoding to better reflect periodic behaviour.
- For models such as Random Forest and SVR, even 50,000 rows were computationally expensive, so additional resampling was applied to speed up training while preserving performance

Recommendation: Deploy Gradient Boosting for better accuracy. However, there is very minimal gap between few models in terms of R2 metric.

Future Work:

- Due to computational limits, the models were trained with 50K data only. It would be better if the models trained with whole dataset.
- Applying few more optimization techniques during Model training and Hyper parameter tuning to understanding whether those things can improve the model performance

References

- Scikit-learn Developers. (2024). Scikit-learn: Machine Learning in Python. Retrieved from https://scikit-learn.org
- Zhou, Z. H. (2012). *Ensemble methods: Foundations and algorithms*. CRC press.
- Chen, T., He, T., Bengio, Y., Koren, V., & Cortes, C. (2015). XGBoost: Scalable and flexible gradient boosting. from https://xgboost.readthedocs.io
- Course materials such as PPT’s, Collab notebooks and Exercises during class sessions from

- Moodle.
- Feature engineering references (https://github.com/snehapadgaonkar/Household-energy-consumption-prediction/blob/main/README.md)

Additionally, AI tools were used only to assist with plotting the graphs, as I have limited knowledge with that matplotlib library.