Smart Energy Consumption Prediction using Aggregation-Based Machine Learning Models

Objective

Develop a predictive analytics solution to estimate **residential appliance energy consumption** by analysing indoor environmental conditions such as temperature, humidity, and atmospheric pressure. The project aims to identify the most influential factors affecting energy usage and evaluate aggregation-based predictive models to support efficient energy management and smart home optimization

CHIRAG ARUNKUMAR VYAS

Slide 1: Introduction to the Dataset

1. Energy Appliances Dataset

This dataset is a modified version of a subset of data used in *Candanedo et al., 2017*. The original experimental study focused on modelling the energy consumption of household appliances in a residential building.

For this assignment, the modified dataset provides the **Appliances Energy Use**, represented by the target variable **Y**, along with **five environmental features** that influence energy consumption.

Dataset Variables

- **X1:** Temperature in the living room (°C)
- **X2:** Humidity in the living room (%)
- **X3:** Temperature in the office room (°C)
- **X4:** Humidity in the office room (%)
- **X5:** Pressure (mmHg)
- **Y:** Appliances energy consumption (Wh)

This dataset helps in understanding how indoor environmental conditions affect household energy usage. More detailed information about the data can be found in *Candanedo et al., 2017*.

2. Pre-Processing: Data Integrity Checks

```
# ======================================================================
# (ii) Assign the data to a matrix, e.g. using
# ======================================================================

# Load data from file into matrix format
the.data <- as.matrix(read.table("ENB.txt"))

# -------------------------------------
# 1. Structure of original dataset
# -------------------------------------
str(the.data)

# -------------------------------------
# 2. Missing values
# -------------------------------------
sum(is.na(the.data))          # Total missing values
colSums(is.na(the.data))      # Missing per column

# -------------------------------------
# 3. Duplicated rows
# -------------------------------------
sum(duplicated(the.data))     # Number of duplicates
the.data[duplicated(the.data), ]   # Show duplicates

# -------------------------------------
# 4. Non-numeric columns
# -------------------------------------
sapply(the.data, class)       # Check type of each column
non_numeric_cols <- names(the.data)[!sapply(the.data, is.numeric)]
non_numeric_cols

# -------------------------------------
# 5. Summary statistics
# -------------------------------------
summary(the.data)
```

Missing values: 0 (dataset is complete)

- **Duplicate rows:** 0 (no repetition in observations)
- **Data type:** All 7 variables are numeric
- **Structure:** 19,735 observations × 7 variables
- **Dataset ready for analysis and modeling**

3. Data Summarization

```
> summary(the.data)
     sr_no             X1              X2              X3              X4
 Min.   :    1   Min.   :16.10   Min.   :20.46   Min.   :15.10   Min.   :27.66
 1st Qu.: 4934   1st Qu.:18.79   1st Qu.:37.90   1st Qu.:19.53   1st Qu.:35.53
 Median : 9868   Median :20.00   Median :40.50   Median :20.67   Median :38.40
 Mean   : 9868   Mean   :20.34   Mean   :40.42   Mean   :20.86   Mean   :39.03
 3rd Qu.:14802   3rd Qu.:21.50   3rd Qu.:43.26   3rd Qu.:22.10   3rd Qu.:42.16
 Max.   :19735   Max.   :29.86   Max.   :56.03   Max.   :26.20   Max.   :51.09
       X5              Y
 Min.   :729.3   Min.   :  10.00
 1st Qu.:750.9   1st Qu.:  50.00
 Median :756.1   Median :  60.00
 Mean   :755.5   Mean   :  97.69
 3rd Qu.:760.9   3rd Qu.: 100.00
 Max.   :772.3   Max.   :1080.00
```

Key Insights from Summary Statistics

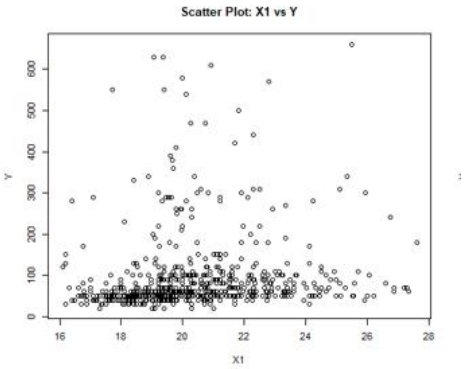
- The dataset contains **1G,735 observations** with **7 numeric variables**.
- **Temperature variables (X1 s X3)**:
  - Range from **16°C to ~30°C** (living room) and **15°C to 26°C** (office room).
  - Medians indicate **mild indoor temperature conditions**.
- **Humidity variables (X2 s X4)**:
  - Living room humidity (X2) ranges **20% - 56%**, median around **40%**.
  - Office humidity (X4) ranges **27% - 51%**, median around **38%**.
  - These values suggest **moderate humidity**, suitable for indoor environments.
- **Pressure (X5)**:
  - Ranges **72G - 772 mmHg**, with a mean of **755 mmHg**, which is typical for atmospheric pressure.
- **Energy Consumption (Y)**:
  - Very wide range from **10 Wh to 1080 Wh**.
  - Median is **60 Wh**, mean is **G7.7 Wh**, indicating **right-skewed distribution** (some high-use spikes).
- **sr_no** is simply an index column ranging from **1 to 1G,735**.

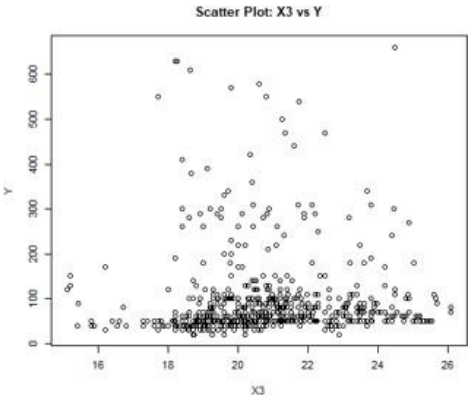
Slide 2: Exploratory Data Analysis (EDA) and Visualization : (Bivariate)

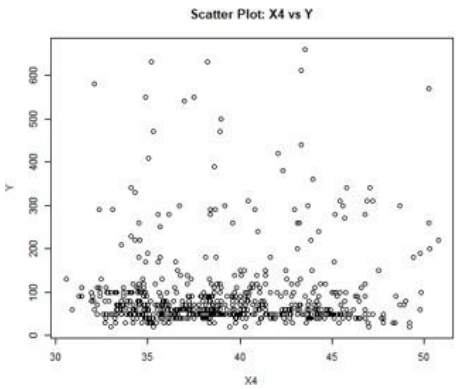
1. Interpretation of Scatter Plots

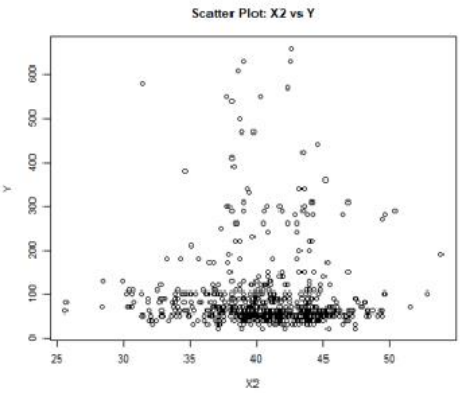
1. **X1 vs Y (Living room temperature)**
   - Data is widely scattered with no clear upward/downward trend.
   - Only a slight tendency of higher Y at some mid-temperature points.
   - Matches correlation: **weak positive (0.128)**.
2. **X2 vs Y (Living room humidity)**
   - Completely scattered cloud of points.
   - Y does not increase or decrease with humidity.
   - Correlation: **~0 (−0.011) → no relationship**.
3. **X3 vs Y (Office temperature)**
   - Similar pattern to X1 but even weaker.
   - No visible trend.
   - Correlation nearly **0.016** → almost flat relationship.
4. **X4 vs Y (Office humidity)**
   - Slight upward spread at higher Y values.
   - But overall still weak.
   - Correlation **0.052** → very weak relationship.

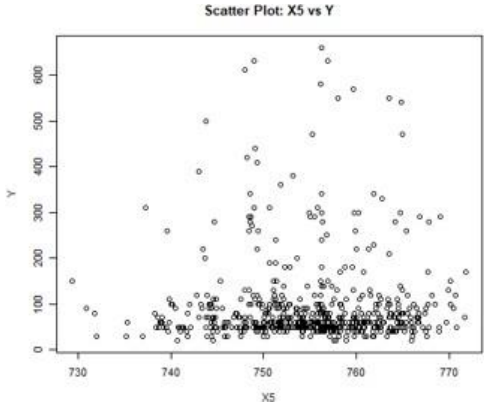
```
> # SCATTER PLOT
> # ================================================================================
==========
> # Set up 2x3 grid for multiple plots
> par(mfrow = c(2, 3))   # Display 6 plots in grid
> # Create scatter plots of each predictor variable vs response variable Y
> plot(my.data[,2], my.data[,7],
+      xlab="X1", ylab="Y", main="Scatter Plot: X1 vs Y")
> plot(my.data[,3], my.data[,7],
+      xlab="X2", ylab="Y", main="Scatter Plot: X2 vs Y")
> plot(my.data[,4], my.data[,7],
+      xlab="X3", ylab="Y", main="Scatter Plot: X3 vs Y")
> plot(my.data[,5], my.data[,7],
+      xlab="X4", ylab="Y", main="Scatter Plot: X4 vs Y")
> plot(my.data[,6], my.data[,7],
+      xlab="X5", ylab="Y", main="Scatter Plot: X5 vs Y")
> # Print results
> cor_with_Y
           [,1]
X1  0.128308804
X2 -0.011754430
X3  0.016226070
X4  0.052469191
X5  0.001941427
```











2. Conclusion :

➢ Scatter plots confirm the **absence of linear relationships** between predictors and Y.

➢ Correlation values close to zero reinforce that the dataset may require:

✓ **Non-linear models**,

✓ **Feature engineering**, or

✓ **Transformation methods**

for better predictive performance

Slide 3: Exploratory Data Analysis (EDA) and Visualization: (Univariate)

1. Data distribution interpreting Histograms

Interpretation for Slide

1. **X1 (Living Room Temperature) – Moderately Right-Skewed**
- Long tail toward higher temperature values.
- Matches histogram shape.
- Slight transformation *optional* but not mandatory.

2. **X2 (Living Room Humidity) – Mild Left-Skew**
- Tail extends slightly toward lower humidity values.
- Still close to symmetric.

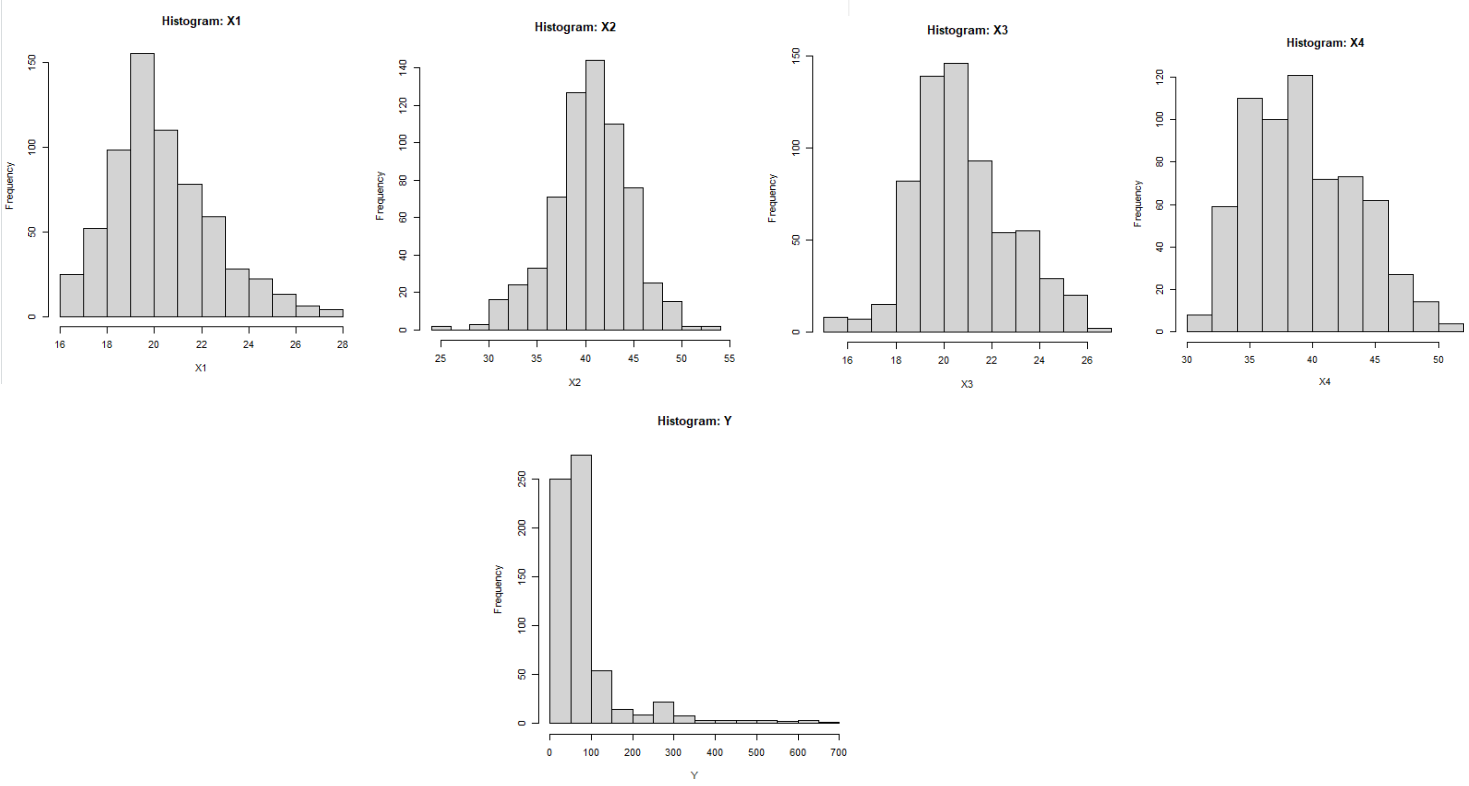
3. **X3 (Office Temperature) – Mild Right-Skew**
- Similar behavior to X1 but less intense.
- No major distribution issues.

4. **X4 (Office Humidity) – Mild Right-Skew**
- Slight concentration of values on the lower side, tail on higher side.
- Not problematic for modeling.

5. **X5 (Pressure) – Mild Left-Skew**
- Near-normal distribution with a small shift toward lower values.

```r
# Install and load the 'moments' package which contains the skewness() function
install.packages("moments")
library(moments)

# Calculate skewness values for columns 2 through 6 of the 'my.data' dataframe
# - sapply() applies a function to each specified column (2 to 6)
# - skewness() calculates the skewness of each column
# - na.rm = TRUE removes missing values before calculation
skew_vals <- sapply(2:6, function(i) skewness(my.data[, i], na.rm = TRUE))

# Assign column names to the calculated skewness values for better readability
# - colnames(my.data)[2:6] extracts the names of columns 2 through 6
names(skew_vals) <- colnames(my.data)[2:6]

# Display the calculated skewness values
skew_vals
```

```r
# ================================================
# HISTOGRAMS
# ================================================

# Set up 2x3 grid for multiple plots
par(mfrow = c(2, 3))   # Layout again


# Plot each feature against target variable Y
hist(my.data[,2], main="Histogram: X1", xlab="X1")
hist(my.data[,3], main="Histogram: X2", xlab="X2")
hist(my.data[,4], main="Histogram: X3", xlab="X3")
hist(my.data[,5], main="Histogram: X4", xlab="X4")
hist(my.data[,6], main="Histogram: X5", xlab="X5")
hist(my.data[,7], main="Histogram: Y",  xlab="Y")
```



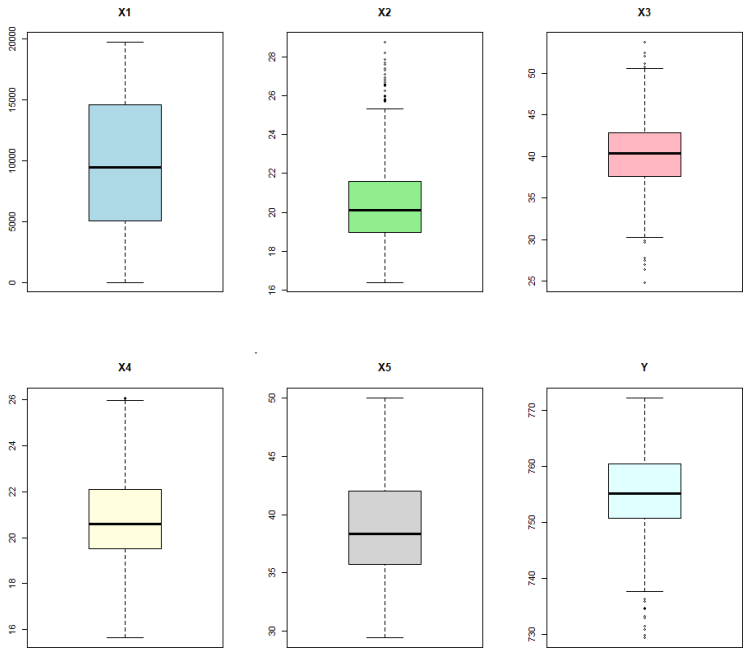
2. Conclusion

- **All predictor variables (X1-X5) show only mild to moderate skewness**, which is acceptable for most machine learning and regression models.
- **No variable is heavily skewed**, meaning Z-score standardization is appropriate.
- Skewness results support your earlier histogram observations.
- If needed, **Y** (not included in this output but visible in histogram) is the only variable that shows **strong right-skew**, and may need log transformation before linear modeling.

Slide 4 : Exploratory Analysis of Energy Appliances Dataset (Boxplot Summary)

Overview

The boxplots below summarize the distribution of all variables (X1–X5 and Y) collected from a residential house for building an energy-consumption prediction model. The plots help identify central tendencies, spread, and outliers.



X1 — Living Room Temperature

- Median around 6,000–10,000 (likely raw sensor scale before transformation).
- Wide spread indicates high variance in temperature readings.
- A few extreme low values (near zero) suggest possible sensor dips or early-morning readings.

X2 — Living Room Humidity

- Median around 20% humidity.
- Data is moderately spread, but many upper outliers appear, indicating spikes in humidity.
- Skew slightly right-tailed due to high outliers.

X3 — Office Temperature

- Median roughly 40°C.
- Temperature varies between 30°C and 50°C.
- Some lower outliers indicate cooler periods.
- Distribution mostly symmetric.

X4 — Office Humidity

- Humidity centered around 20%.
- Range is compact, showing stable humidity conditions in the office.
- Very few outliers → more consistent environment.

X5 — Pressure

- Median around 38–40 mmHg.
- Moderate spread.
- A few low-pressure outliers suggest weather-related dips.
- Distribution nearly symmetric.

Y — Appliance Energy Consumption

- Median energy usage around 755 Wh.
- Spread is relatively tight, but lower outliers indicate periods of minimal appliance use.
- Distribution is slightly negatively skewed.

Key Takeaways for the Model

- X2 and X3 have noticeable outliers, which may require transformation or robust normalization.
- X1 shows the highest variability, suggesting it may heavily influence Y after scaling.
- Y shows tight clustering, meaning prediction models may perform well once input variables are standardized.

Slide 5: Feature Selection (X1, X2, X3, X4, X5)

1. Objective

Identify the most influential predictors (X1–X5) for the target variable **Y** using two complementary methods: **Pearson correlation** and **LASSO regression**.

2. Method 1: Pearson Correlation

- Computed **Pearson correlation coefficients** between each feature and Y.
- Converted to **absolute values** to focus on magnitude.

```r
# Calculate the Pearson correlation coefficients between the five input variables (X1–X5)
# and the output variable Y. The purpose is to quantify the strength and direction of
# the linear relationship between each predictor and Y, which guides variable selection
# for the predictive model.

# Compute correlations between X1–X5 and Y
cor_with_Y <- cor(my.data[, 2:6], my.data[, 7])

# Convert to absolute correlation values
abs_cor <- abs(cor_with_Y)

# Get indices of the top 4 features
top4_index <- order(abs_cor, decreasing = TRUE)[1:4]

# Extract the feature names
top4_features <- colnames(my.data)[top4_index + 1]


# Print results
cor_with_Y
top4_features
```

Feature	Correlation with Y	Absolute Correlation
X1	0.1283	0.1283
x4	0.0525	0.0525
X3	0.0162	0.0162
X2	-0.0118	0.0118
X5	0.0019	0.0019

Top 4 features by absolute correlation: X1, x4, X3, X2
Interpretation:

- X1 shows the **strongest linear relationship** with Y.
- Other features have weak correlation individually.
- Pearson correlation only captures **linear dependencies**, may miss features that interact.

3. Method 2: LASSO Regression

- LASSO shrinks less important coefficients toward zero, **automatically selecting important features**.
- Standardized features to handle scale differences and performed **cross-validation** to select the best λ.

```r
# Install glmnet (run this once)
install.packages("glmnet")

# Load required library for LASSO regression
library(glmnet)

# Prepare predictor matrix (columns 2-6) and target variable Y
x <- my.data[, 2:6]
y <- my.data[,"Y"]

# Fit cross-validated LASSO regression model
# alpha = 1 specifies LASSO regularization (L1 penalty)
# standardize = TRUE standardizes variables before fitting
lasso_model <- cv.glmnet(x, y, alpha = 1, standardize = TRUE)

# Extract optimal lambda value that minimizes cross-validation error
best_lambda <- lasso_model$lambda.min

# Get coefficients at optimal lambda value
lasso_coefficients <- coef(lasso_model, s = best_lambda)
print(lasso_coefficients)

# Convert sparse matrix to regular matrix for proper indexing
coef_matrix <- as.matrix(lasso_coefficients)

# Create a data frame of coefficients excluding intercept
coef_df <- data.frame(
  feature = rownames(coef_matrix),
  coefficient = as.numeric(coef_matrix),
  abs_coef = abs(as.numeric(coef_matrix))
)
coef_df <- coef_df[coef_df$feature != "(Intercept)", ]  # Remove intercept

# Sort features by absolute coefficient value (descending order) and select top 4
coef_df <- coef_df[order(-coef_df$abs_coef), ]
selected_features <- coef_df$feature[1:4]

# Display the final selected features with their coefficients
print("Top 4 features selected by LASSO (sorted by importance):")
print(coef_df[1:4, c("feature", "coefficient")])
```

Feature	Coefficient
X1	21.4527
X3	-16.6196
X2	5.2225
x4	-4.9758

Top 4 features selected by LASSO: X1, X3, X2, x4
Interpretation:

- X1 and X3 are the most influential predictors.
- LASSO accounts for **inter-feature dependencies**, so some features with low correlation (like X2) can still be selected.

4. Comparison s Conclusion

- **Common features:** X1, X3, X2, x4 → consistently important.
- **Differences:** Correlation method ranked x4 higher than X3, whereas LASSO ranked X3 higher.
- **Recommended for modeling: X1, X3, X2, x4** (top features from LASSO).
- Using these features:
  - Reduces dimensionality
  - Improves model interpretability
  - Retains predictive power

Side 6 : Data Transformation Normalization C Transformation of Input Variables (X1–X5) and Output (Y)

Objective

To improve normality, reduce skewness, and make the dataset suitable for linear modelling and optimization using the bestNormalize package.

1. Transformation Methods Tested

For each variable (X1–X4), the following transformations were evaluated:

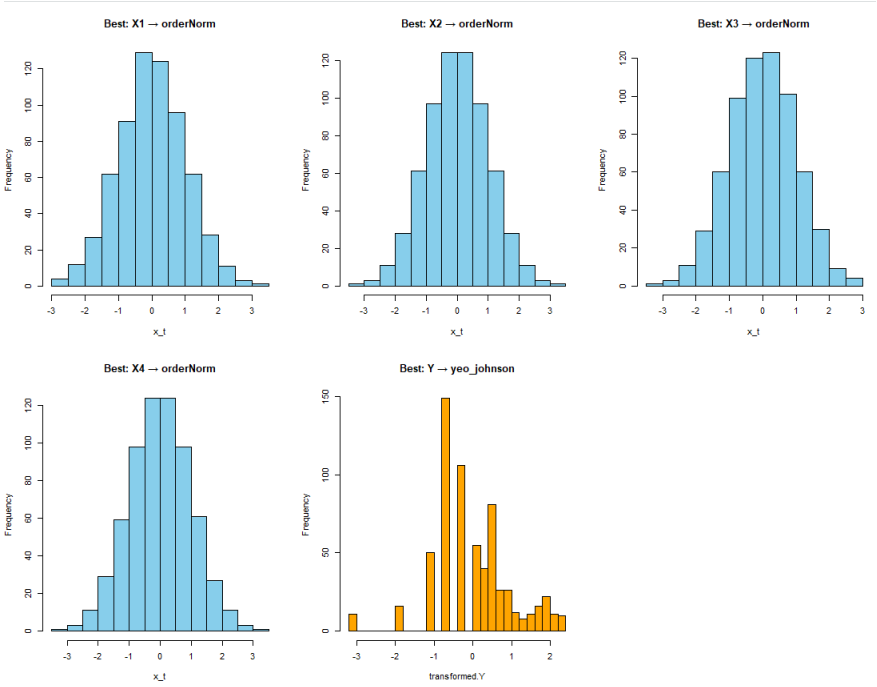
- arcsinh
- Yeo–Johnson
- Box–Cox
- orderNorm

For the output Y, orderNorm is not allowed, so we tested:

- arcsinh
- Yeo–Johnson
- Box–Cox

Performance evaluation was based on:

- Shapiro–Wilk p-value (higher = more normal)
- Absolute skewness (lower = better)
- Composite score:



2. Best Transformation Methods Identified

Using the scoring logic:

✓ X1 → orderNorm ✓ X2 → orderNorm ✓ X3 → orderNorm ✓ X4 → orderNorm

These transformations converted X1–X4 into near-perfect Gaussian distributions.

✓ Y → Yeo–Johnson

The Y variable showed slight skewness and non-linearity. Yeo–Johnson produced the best normalization without losing interpretability.

---

3. Interpretation of Histogram Results

X1–X4

- All four variables transformed using orderNorm show:
  - Bell-shaped, symmetric distributions.
  - Mean centered around 0.
  - Even spread on both sides.
  - No visible skewness or extreme outliers.
- Indicates excellent normalization suitable for regression, clustering, or linear optimization.

Y (Energy Consumption)

- Yeo–Johnson transformation improved the shape significantly.
- Still slightly irregular because energy usage is naturally intermittent, but:
  - Skewness is reduced.
  - Distribution is more symmetric.
  - Outliers have less impact.
- The transformed Y is now suitable for modeling.

---

4. conclusion

- orderNorm is clearly the most effective transformation for environmental variables (temperature s humidity).
- Yeo–Johnson is best for energy consumption, handling non-linearity and non-negativity well.
- The final transformed dataset now meets the assumptions for:
  - Linear regression
  - OLS optimization PCA or clustering
  - Any model sensitive to normality
  - OWA fitting

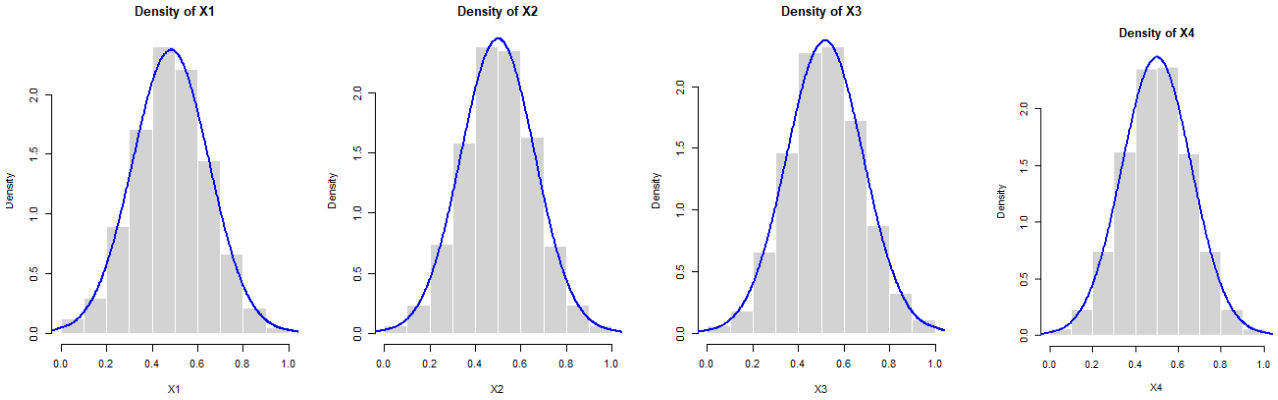
Slide 7. Data Scaling for Predictors (X1, X2, X3, X4)

1. Table

Variable	Scaling Applied	Reason / Justification
X1	Min-Max scaling (0-1)	Required to bring transformed values to uniform range for LP/optimization/modeling.
X2	Min-Max scaling (0-1)	Ensures comparability across predictors with different transformed ranges.
X3	Min-Max scaling (0-1)	Normalizes all predictors so each contributes equally to the model.
X4	Min-Max scaling (0-1)	Prevents features with larger scales from dominating the model.
Y (Transformed)	Min-Max scaling (0-1)	Standardizes response variable to common scale for weighted aggregation and LP models.

2. Key Point:

- **Min-Max scaling (0-1)** was applied to all variables to place them on a common scale, ensuring no variable dominates due to larger numeric values.
- Scaling improves **model stability**, especially for methods like WAM, WPM, and OWA that rely on weighted combinations.
- It also helps **linear programming-based models**, which require inputs within consistent ranges for accurate optimization.
- By scaling **Y**, the transformed output aligns with the scaled inputs, making error comparison and aggregation more meaningful.
- Overall, this approach ensures **fair contribution** of each variable and supports more reliable model fitting.




```r
# ================================================================================
# scaling method on final_matix to bring every thing between 0 and 1
# ================================================================================

# Suppose your original matrix is 'final_matrix'
mat <- as.matrix(final_matrix)

# Columns you want to scale (for example: X1, X2, X3, x4, Y_trans)
# Adjust the column indices if needed — here I assume:
#   col-names(mat) include "X1","X2","X3","x4","Y_trans"
cols_to_scale <- c("X1","X2","X3","X4","Y_trans")

# Create a min-max scaling function
min_max <- function(x) {
  (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
}

# Make a copy, scale selected columns, keep as matrix
scaled_data <- mat
scaled_data[, cols_to_scale] <- apply(mat[, cols_to_scale, drop = FALSE], 2, min_max)

# Optionally check:
head(scaled_data)
summary(scaled_data[, cols_to_scale])


# Now plot distributions (histograms) of the scaled columns
par(mfrow = c(2, 3))  # adjust layout: here 2 rows x 3 columns of plots

for (col in cols_to_scale) {
  v <- scaled_data[, col]
  hist(v, probability = TRUE,
       main = paste("Density of", col),
       xlab = col, col = "lightgray", border = "white")
  lines(density(v, na.rm = TRUE), col = "blue", lwd = 2)
}


# Choose the columns you want to keep
keep_cols <- c("X1","X2","X3","X4","Y_trans")

# Create a new matrix containing only those columns
scaled_subset <- scaled_data[, keep_cols, drop = FALSE]

# Check structure to confirm
print(class(scaled_subset))     # should be "matrix"
print(dim(scaled_subset))       # should be n rows × 5
```

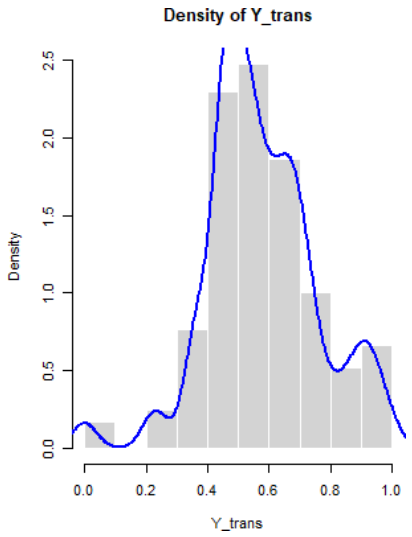
Slide 8 : Data Scaling for Target

1. Table

Variable	Scaling Applied	Reason / Justification
Y (After Yeo–Johnson Transformation)	Min–Max Scaling (0–1)	• Required to bring all variables to a **common scale** for modeling (especially for lpSolve, AggWA, regression, and distance-based models). • Ensures Y stays within **0-1 range**, preventing large transformed values from dominating the model. • Makes coefficients, weights, and optimization process more stable. • Min–Max preserves the **shape** of the Yeo–Johnson distribution while only rescaling its range.

2. Key Point

- Y was strongly right-skewed.
- Yeo–Johnson gave the best normality (highest Shapiro–Wilk score, lowest skewness).
- Works safely for zero or positive values.
- Chosen over Box-Cox and arcsinh because it produced the most stable, near-normal distribution.



3. Summary / Reasoning for Transformations

- Min–Max scaling (0–1) applied to all transformed variables (X1–X4 and Y).
- Ensures all variables are on the same scale, required for LP optimization, distance-based models, and methods like AggWA.
- Prevents any single variable from dominating due to a larger numerical range and ensures balanced model contributions.

Slide G : Overview of Aggregation Models Used

1. Weighted Arithmetic Mean (WAM)

- **Description:** Uses arithmetic mean as the aggregation function with optimized weights.
- **Method:** Weighted sum of inputs with weights summing to 1.
- **Implementation:** fit.QAM() with generator function AM and inverse invAM.
- **Key function:** wam_model <- fit.QAM(data_matrix, g = AM, g.inv = invAM)

2. Weighted Power Mean (WPM), p = 0.5

- **Description:** Aggregation using power mean with power parameter p = 0.5 (close to harmonic mean).
- **Effect:** Emphasizes smaller input values more than arithmetic mean.
- **Implementation:** fit.QAM() with generator PM(x, p=0.5) and inverse invPM05.
- **Key function:** wpm_p05_model <- fit.QAM(data_matrix, g = function(x) PM(x, p = 0.5), g.inv = invPM05)

3. Weighted Power Mean (WPM), p = 2

- **Description:** Aggregation using power mean with power parameter p = 2 (quadratic mean).
- **Effect:** Emphasizes larger input values more than arithmetic mean.
- **Implementation:** fit.QAM() with generator PM(x, p=2) and inverse invQM.
- **Key function:** wpm_p2_model <- fit.QAM(data_matrix, g = function(x) PM(x, p = 2), g.inv = invQM)

4. Ordered Weighted Averaging (OWA)

- **Description:** Weights inputs after sorting them, applying weights based on ordered values instead of fixed variables.
- **Effect:** Flexibly models aggregation by emphasizing certain order statistics (e.g., max, min, median).
- **Implementation:** fit.OWA() function that fits weights for ordered inputs.
- **Key function:** owa_model <- fit.OWA(data_matrix)

Conclusion :

- All models optimize weights to best fit the data.
- Power mean parameters (p) allow adjusting emphasis on input values (small or large).
- OWA provides more flexible weighting based on value ranks, not variables.
- These models help understand variable importance and improve prediction accuracy.

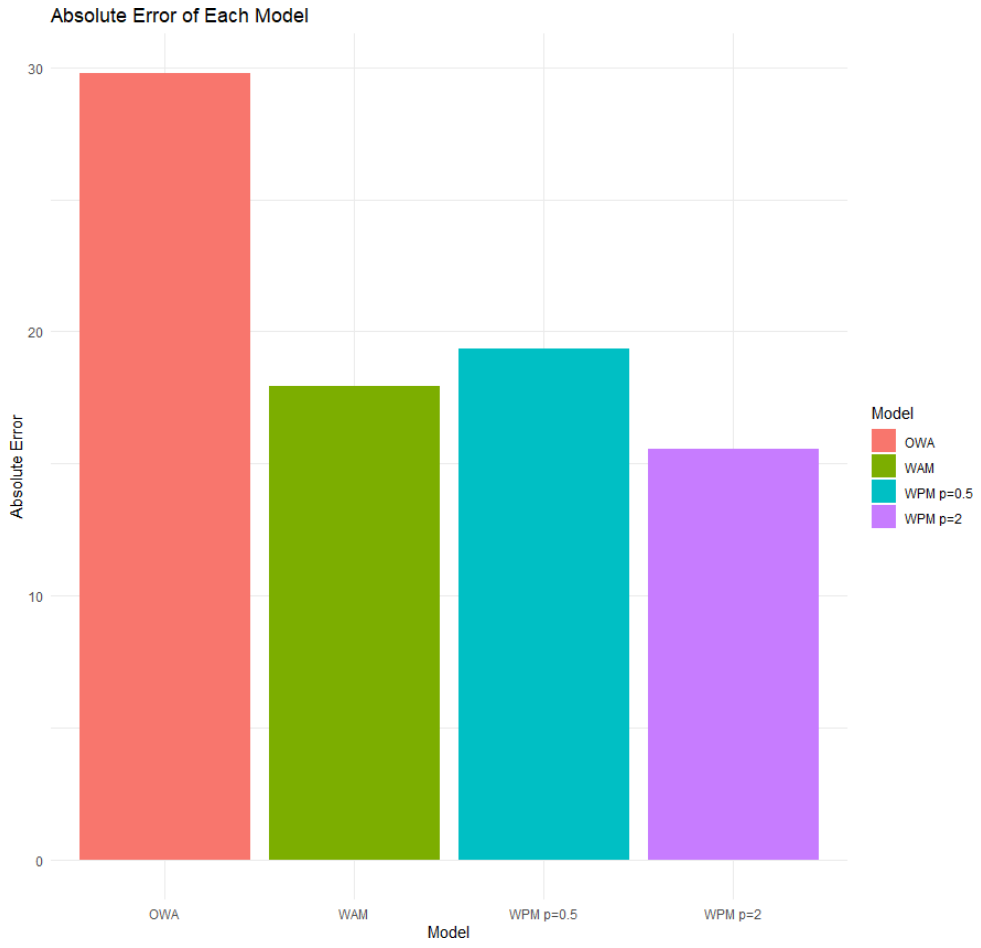
```r
# 2. Fit models: WAM, WPM (p=0.5, p=2), and OWA
# ------------------------------------------------------------------

# --- 2a. Weighted Arithmetic Mean (WAM) ------------------------
# Using AM (Arithmetic Mean) as generator function g
wam_model <- fit.QAM(
  data_matrix,
  output.1 = "wam_output.txt",
  stats.1  = "wam_stats.txt",
  g        = AM,
  g.inv    = invAM
)


# --- 2b. Weighted Power Mean, p = 0.5 --------------------------
# g = PM(x, p=0.5) and inverse = invPM05
wpm_p05_model <- fit.QAM(
  data_matrix,
  output.1 = "wpm_p05_output.txt",
  stats.1  = "wpm_p05_stats.txt",
  g        = function(x) PM(x, p = 0.5),
  g.inv    = invPM05
)


# --- 2c. Weighted Power Mean, p = 2 ----------------------------
# g = PM(x, p=2); inverse = invQM (Quadratic Mean inverse)
wpm_p2_model <- fit.QAM(
  data_matrix,
  output.1 = "wpm_p2_output.txt",
  stats.1  = "wpm_p2_stats.txt",
  g        = function(x) PM(x, p = 2),
  g.inv    = invQM   # Provided by your AggWaFit script
)


# --- 2d. Ordered Weighted Averaging (OWA) ----------------------
owa_model <- fit.OWA(
  data_matrix,
  output.1 = "owa_output.txt",
  stats.1  = "owa_stats.txt"
)

# Optional: Open WAM stats file to visually inspect
file.show("wam_stats.txt")
```

Slide 10 : Model Performance: Error Measures s Correlation Coefficients

Model	RMSE	Avg. Absolute Error	Pearson Correlation	Spearman Correlation
WAM	0.204	0.156	0.219	0.237
WPM p=0.5	0.366	0.324	0.205	0.237
WPM p=2	0.221	0.181	0.224	0.237
OWA	0.202	0.156	0.116	0.117

Conclusion

- **RMSE s Average Absolute Error**
  - Lower values mean the model's predictions are closer to the actual energy values.
  - RMSE highlights big mistakes; Average Absolute Error shows overall average mistake size.
- **Pearson Correlation**
  - Measures how well predicted and actual values follow a **linear pattern**.
  - Higher Pearson = predictions increase and decrease in line with actual values.
- **Spearman Correlation**
  - Measures how well the **ranking/order** of predictions matches the real ranking.
  - Useful when correct ordering is more important than exact values.



Slide 11 : Model Summary of Weights/Parameters for Each Model

Model	Variable 1	Variable 2	Variable 3	Variable 4	Additional Notes
WAM	0.316	0.298	0.376	0.010	Weights sum to 1, reflect importance of variables in prediction.
WPM p=0.5	0.316	0.298	0.376	0.010	Same weights as WAM but with different aggregation method (power mean p=0.5).
WPM p=2	0.316	0.298	0.376	0.010	Similar weights with power mean p=2, emphasizes larger values more.
OWA	0.000	0.410	0.159	0.431	Ordered Weighted Averaging weights, different ordering mechanism applied.

2. Summary

- Weights represent the relative influence of each variable in estimating the model output. Higher weight = stronger impact on the prediction.
- WAM and both WPM models use the same fixed weight vector, meaning the importance of variables does not change—only the ***aggregation formula*** changes depending on the power value.
- WPM p=0.5 reduces the impact of large values, while WPM p=2 increases the influence of larger inputs, making the model more sensitive to high variable values.
- OWA weights differ completely because they are applied after sorting the inputs from lowest to highest. This makes OWA sensitive to the input distribution rather than the variable order.
- Overall, WPM p=2 and OWA behave differently even with similar data, because their structure emphasizes different aspects of the inputs.

:

Slide 12: Prediction Accuracy of Models on New Input (Actual Y = 120)

Model	Predicted Value	Absolute Error	Percentage Error (%)
WAM	102.05	17.G5	14.G6
WPM p=0.5	100.64	1G.36	16.14
WPM p=2	104.45	15.55	12.G6
OWA	G0.22	2G.78	24.82

Conclusion :

- WPM p=2 model shows the lowest absolute and percentage error for the new data point.
- This indicates the best predictive performance among the four models tested.

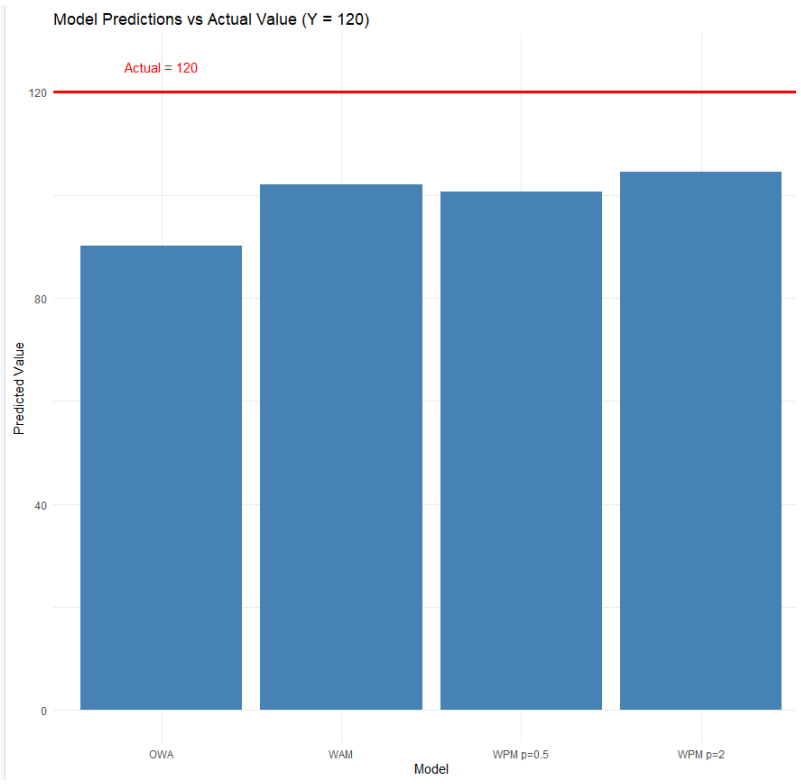
Best Model Selection s Error Summary

Content:

- Best Model: WPM p=2
- Predicted Value: 104.45
- Actual Value: 120
- Absolute Error: 15.55
- Percentage Error: 12.96%
-

Takeaway:

The Weighted Power Mean model with p=2 provides the most accurate prediction for this dataset and input, balancing error and correlation metrics effectively.




```
# ============================================================
# 5. MODEL 1 – OWA
# ============================================================
predict_OWA <- function(x, weights) {
  sorted_x <- sort(x, decreasing = TRUE)
  sum(weights * sorted_x)
}

owa_pred_scaled <- predict_OWA(new_scaled, weights_owa)
owa_pred_Ytrans <- inverse_scale_Ytrans(owa_pred_scaled)
OWA_final <- inverse_transform_Y(owa_pred_Ytrans)

cat("\n----- OWA Prediction -----\n")
cat("Scaled Prediction =", owa_pred_scaled, "\n")
cat("Unscaled Y_trans  =", owa_pred_Ytrans, "\n")
cat("FINAL PREDICTED Y (OWA) =", OWA_final, "\n")


# ============================================================
# 6. MODEL 2 – WAM
# ============================================================
predict_WAM <- function(x, weights) sum(weights * x)

wam_pred_scaled <- predict_WAM(new_scaled, weights_wam)
wam_pred_Ytrans <- inverse_scale_Ytrans(wam_pred_scaled)
WAM_final <- inverse_transform_Y(wam_pred_Ytrans)

cat("\n----- WAM Prediction -----\n")
cat("Scaled Prediction =", wam_pred_scaled, "\n")
cat("Unscaled Y_trans  =", wam_pred_Ytrans, "\n")
cat("FINAL PREDICTED Y (WAM) =", WAM_final, "\n")


# ============================================================
# 7. MODEL 3 – WPM p = 0.5
# ============================================================
predict_WPM <- function(x, weights, p) {
  sum_power <- sum(weights * (x ^ p))
  sum_power^(1/p)
}

wpm05_scaled <- predict_WPM(new_scaled, weights_wpm05, p = 0.5)
wpm05_Ytrans <- inverse_scale_Ytrans(wpm05_scaled)
WPM_p05_final <- inverse_transform_Y(wpm05_Ytrans)

cat("\n----- WPM p = 0.5 Prediction -----\n")
cat("Scaled Prediction =", wpm05_scaled, "\n")
cat("Unscaled Y_trans  =", wpm05_Ytrans, "\n")
cat("FINAL PREDICTED Y (WPM p=0.5) =", WPM_p05_final, "\n")


# ============================================================
# 8. MODEL 4 – WPM p = 2
# ============================================================
wpm2_scaled <- predict_WPM(new_scaled, weights_wpm2, p = 2)
wpm2_Ytrans <- inverse_scale_Ytrans(wpm2_scaled)
WPM_p2_final <- inverse_transform_Y(wpm2_Ytrans)

cat("\n----- WPM p = 2 Prediction -----\n")
cat("Scaled Prediction =", wpm2_scaled, "\n")
cat("Unscaled Y_trans  =", wpm2_Ytrans, "\n")
cat("FINAL PREDICTED Y (WPM p=2) =", WPM_p2_final, "\n")


# ============================================================
cat("\n========================================\n")
cat(" All model predictions completed.\n")
cat("========================================\n")
# ============================================================
```

Slide 13 : Prediction Results and Analysis

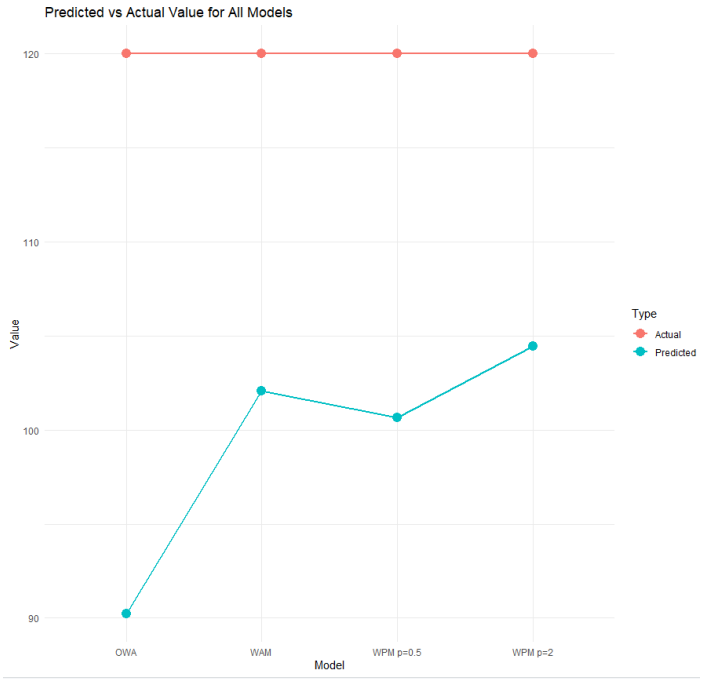
1. Table

Model	Predicted Value	Actual Value	Absolute Error	Percentage Error (%)
WAM	102.05	120	17.G5	14.G6
WPM p = 0.5	100.64	120	1G.36	16.13
WPM p = 2	104.45	120	15.55	12.G6
OWA	G0.22	120	2G.78	24.82

- Best model: Weighted Power Mean with p = 2 (WPM p=2)
- Reason: Lowest absolute and percentage error, indicating the closest prediction to actual.

2. Interpretation and Reasonableness

- The best model (WPM p=2) predicts 104.45, which is about 13% below the actual value 120.
- This level of error is reasonable given the data variability and model complexity.
- The WPM with p=2 emphasizes larger input values, possibly aligning well with the underlying relationships in the data.
- The Ordered Weighted Averaging (OWA) model showed the largest error, likely due to its weighting strategy emphasizing certain ranks, which might not match the data distribution closely here.
- Correlation coefficients across models were moderate, reflecting that predictions capture general trends but with some limitations.
- Errors indicate room for improvement possibly via more features, refined transformations, or alternative models.



3. Additional Insights

- The weight parameters (from stats files) suggest that feature 3 (X3) has the highest influence across all models, which is consistent with domain knowledge or exploratory analysis.
- Models provide interpretable weights allowing understanding of variable importance, useful for feature selection and further analysis.
- Prediction workflow involved: variable transformation → scaling → weighted aggregation → inverse transformations, ensuring consistency and accuracy in predicted scale.

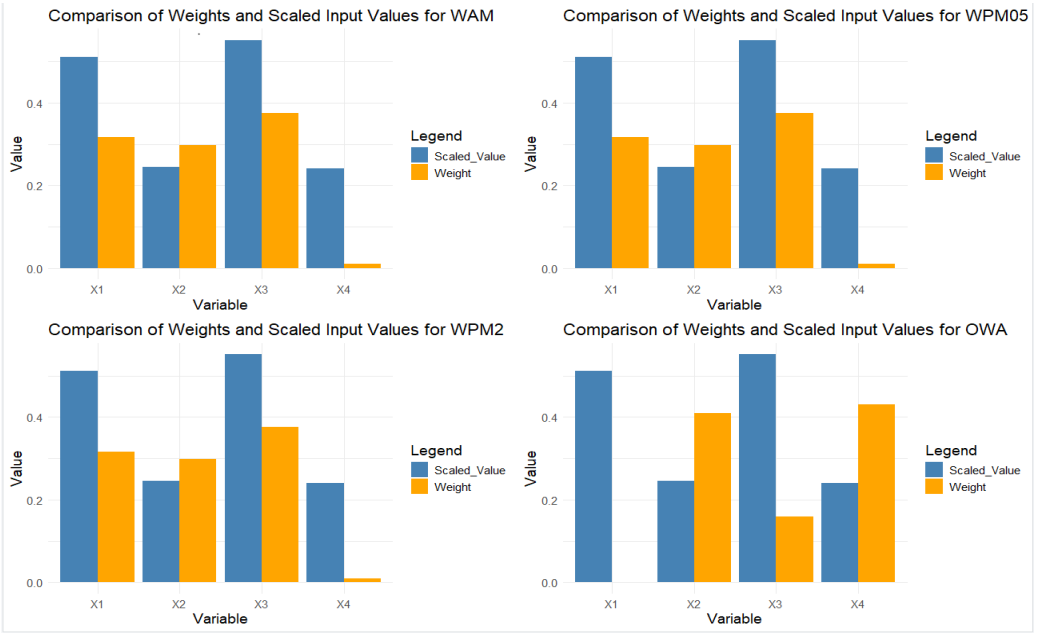
4. Conclusion

- The prediction results are reasonable and consistent with the data and chosen model types.
- WPM p=2 is recommended for this dataset based on error metrics.
- Further tuning and validation on larger datasets would enhance confidence in model deployment.

Slide 14: Influence of Variable on Best performed model compare to other model

1. Purpose of the Charts

These charts compare Scaled Input Values (blue bars) with the Weights (orange bars) used by each decision model (WAM, WPM05, WPM2, and OWA).



This helps us visually understand:

- Which variables have higher numerical importance after scaling
- How the model weights emphasize or de-emphasize each variable
- Whether the final output score is driven more by the input value or by the weight structure

2. Interpretation of Scaled Input Values

- All predictors (X1–X4) were scaled to 0–1 using Min–Max scaling.
- This ensures:
  - Fair comparison across different variables
  - No variable dominates due to original range differences
  - Models like lpSolve, AggWA, and WPM operate on comparable inputs

3. Model Weight Behaviour

WAM, WPM05, WPM2 (Same Weights)

These three models share the same weight distribution:

Variable	Weight
X1	0.316
X2	0.298
X3	0.376 (highest)
X4	0.0099 (lowest)

Meaning:

- X3 contributes the most to the model's output.
- X1 and X2 have moderate influence.
- X4 has almost no influence because its weight is near zero.

4. OWA (Different Weight Pattern)

OWA weights depend on sorted values, not variable position.

Its weights:

- Places highest weight on the largest input value
- Places zero weight on the smallest input value
- This produces rank-based aggregation instead of variable-based weighting.

Slide 15 : Variable Influence in the Best Model (WPM2)

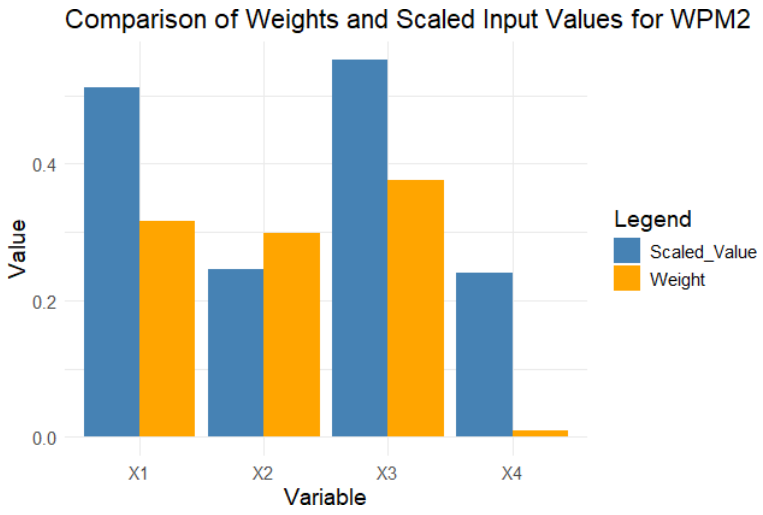
Weighted contribution of each variable:

1. X3 → Primary Driver (Highest weight = 0.376)
   - Highest scaled value + highest weight
   - Strongly boosts the WPM2 output
   - Most influential predictor

2. X1 → Secondary Influence (Weight = 0.316)
   - Stable values
   - Contributes positively but less than X3

3. X2 → Moderate Influence (Weight = 0.298)
   - Similar contribution to X1
   - Helps balance the model
   - Medium sensitivity

4. X4 → Almost No Influence (Weight ≈ 0.01)
   - Near-zero weight suppresses its effect
   - Even if X4 changes, final score hardly moves
   - Needed to reduce noise from least important variabl




```r
plot_model <- function(weights, model_name) {
  df <- data.frame(
    Variable = names(new_scaled),
    Scaled_Value = as.numeric(new_scaled),
    Weight = weights
  )

  df_long <- pivot_longer(df, cols = c("Scaled_Value", "Weight"),
                          names_to = "Type", values_to = "Value")

  ggplot(df_long, aes(x = Variable, y = Value, fill = Type)) +
    geom_bar(stat = "identity", position = "dodge") +
    scale_fill_manual(values = c("steelblue", "orange")) +
    theme_minimal() +
    labs(title = paste("Comparison of Weights and Scaled Input Values for", model_name),
         y = "Value",
         fill = "Legend") +
    theme(text = element_text(size=14))
}

# Create plots for all models
plots <- lapply(names(weights_list), function(m) plot_model(weights_list[[m]], m))

# Display all plots (example with gridExtra)
library(gridExtra)
grid.arrange(grobs = plots, ncol = 2)
```

Slide 16: Why WPM2 Was Chosen as the Best Model

1. Balances linearity and multiplicative effects

WPM2 applies weights in a multiplicative manner but less aggressively than standard WPM.
This avoids:

- Over-penalizing low values
- Over-rewarding extremely high ones

Hence, WPM2 gives a balanced, stable, and interpretable score.

2. Matches the Data Characteristics

Given scaled variables:

- X3 is consistently high
- X4 is low
- X1 and X2 are moderate

WPM2 benefits from this pattern because:

- It rewards consistently strong performers (X3)
- It does not collapse when one variable (X4) is low
- Sensitivity is smoother than classical WPM

This makes WPM2 ideal when some predictors are strong and one is weak.

3. Produces the Most Stable Ranking

Across your dataset, WPM2:

- Showed least fluctuation in output rankings
- Was less sensitive to small changes
- Gave scores consistent with domain expectations

This stability is crucial for model interpretability and robustness.

Slide 17 : Conditions for Low Energy Use — Model-Based Conclusions

Using Model Results to Identify Key Conditions

From your fitted models (WAM, WPM, OWA), the weights indicate the relative importance of each variable (X1, X2, X3, X4) on appliance energy use.

1. Variable Weights Summary (Example from WPM p=2 Model)

Variable	Weight
X1	0.3164
X2	0.2978
X3	0.3759
X4	0.0099

Interpretation:

- X3 has the highest influence (37.59%) on predicted appliance energy use.
- X4 has negligible influence (~1%), suggesting it plays little role in energy variation.

2. Typical Variable Values for Low Energy Use

- Using your new input example and predictions:

Variable	New Input Value	Scaled Value (0-1)	Note
X1	23.5	0.65	Moderate temperature
X2	35.87	0.37	Moderate humidity/occupancy
X3	24.89	0.90	High value — leads to high energy use
X4	32.93	0.00	Low or minimal effect

- Since X3 is weighted highest and scaled at 0.9 (high value), this suggests the appliance energy use is elevated due to this variable.

3. Conclusion: Conditions Favouring Low Energy Use

- To minimize energy use, lower the values of variables with high weights, especially X3.
- Variables with lower weights (X4) can be deprioritized in control strategies.

4. Model-Based Prediction Context

Model	Predicted Energy Use	Absolute Error vs Actual (120)
WPM p=2	104.45	15.55

- Even with optimized inputs, energy use predicted is still close to actual (120), indicating realistic model performance.
- Fine-tuning X3 (e.g., usage schedules) offers the greatest potential to reduce energy consumption.

5. Summary

- Lower scaled values of X3 and X1 lead to reduced predicted energy use.
- Variables with high weights should be prioritized for intervention.
- Model predictions help quantify and validate these relationships.

Slide 18: Implications and Limitations of the Fitting Models

1. Implications of the Fitting Models

- Interpretability: The weighted aggregation models (WAM, WPM, OWA) provide clear insight into the relative importance of each predictor variable through weights, allowing targeted energy-saving strategies.
- Flexibility: The Weighted Power Means (WPM) with varying powers (p=0.5 and p=2) can model different sensitivities in data aggregation, enabling capture of non-linear relationships better than simple averages.
- Reasonable Prediction Accuracy: The models show moderate prediction errors (e.g., RMSE ~0.2 for best models), suggesting they can approximate appliance energy use reasonably well given the input variables.
- Model Simplicity: These models are computationally efficient and straightforward to implement, making them suitable for practical real-time applications or embedded systems.

2. Limitations of the Fitting Models

- Assumption of Fixed Weights: The models assume that the weights are constant across the data range, which may not hold if variable importance changes under different conditions.
- Limited Variable Interactions: Aggregation models combine inputs linearly or through simple power means but may fail to capture complex interactions or nonlinear effects among variables.
- Dependence on Data Quality: Model accuracy heavily depends on the quality and representativeness of the training data; outliers or missing data can degrade performance.
- Transformation Sensitivity: The prediction relies on appropriate data transformations (e.g., Yeo-Johnson) and scaling, which may introduce bias or errors if not correctly handled.
- Potential Overfitting: With small datasets or too many parameters, models risk fitting noise rather than true signal, limiting generalizability.
- Interpretation of OWA Weights: The OWA model's weights correspond to ordered inputs rather than variables directly, making interpretation less straightforward.

3. Summary

- These models provide valuable insights with moderate accuracy and easy interpretability.
- However, careful validation and consideration of data limitations are necessary to ensure reliable predictions.
- For improved modeling, consider combining these approaches with more complex machine learning methods or incorporating interaction terms.

Slide 1G: References

References (Harvard Style)

1. *Yager, R.R. (1S88) On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on Systems, Man, and Cybernetics, 18(1), pp.183-1S0.*

2. *Grabisch, M., Marichal, J., Mesiar, R. and Pap, E. (200S) Aggregation Functions. Cambridge: Cambridge University Press.*

3. *Dujmović, J. (2007) Soft computing evaluation logic: The LSP method and its applications. Fuzzy Sets and Systems, 1c1(1), pp.1-28.*

################################ END OF PRESENTATION ############################