SUPPORT VECTOR REGRESSION WITH GAUSSIAN KERNEL FUNCTION

Machine Learning for
Chemical Engineers
CHE F315

Prof Ajaya Kumar
Pani

Haryaksh Manuh Bhardwaj
2022B4A10900G

Introduction

x1	x2	x3	x4	x5	x6	x7	x8	y
540	0	0	162	2.5	1040	676	28	79.99
540	0	0	162	2.5	1055	676	28	61.89
332.5	142.5	0	228	0	932	594	270	40.27
332.5	142.5	0	228	0	932	594	365	41.05
198.6	132.4	0	192	0	978.4	825.5	360	44.3
266	114	0	228	0	932	670	90	47.03
380	95	0	228	0	932	594	365	43.7
380	95	0	228	0	932	594	28	36.45
266	114	0	228	0	932	670	28	45.85
475	0	0	228	0	932	594	28	39.29
198.6	132.4	0	192	0	978.4	825.5	90	38.07
198.6	132.4	0	192	0	978.4	825.5	28	28.02
427.5	47.5	0	228	0	932	594	270	43.01
190	190	0	228	0	932	670	90	42.33
304	76	0	228	0	932	670	28	47.81
380	0	0	228	0	932	670	90	52.91
139.6	209.4	0	192	0	1047	806.9	90	39.36

- Model: Support Vector Regression (SVR) with Gaussian Kernel
- Advanced regression technique that handles non-linear relationships
- Robust to outliers compared to ordinary least squares methods
- Powerful for complex high-dimensional problems

- Dataset Details:
- 1000 observations with 8 input features (x1-x8) and 1 target variable (y)
- Features represent various process parameters with different scales and ranges
- Target variable appears to represent a continuous outcome (e.g., quality metric, yield)

- Goal:
- Develop an accurate predictive model for precise estimation of target values
- Create a generalizable model that performs well on unseen data
- Establish a systematic approach for parameter optimization in SVR models
-
- Challenge:
- Finding optimal hyperparameters that balance model complexity and generalization
- Avoiding overfitting while capturing underlying patterns in the data

SVR Fundamentals

- Support Vector Regression: Extension of SVM for continuous target variables
  - Uses ε-insensitive loss function: ignores errors within ε distance of true value
  - Creates a "tube" around the regression function where errors are not penalized
  - Only points outside the tube become support vectors and influence the model
- Mathematical Formulation:
  - Objective: minimize $||w||^2 + C \times$ (sum of slack variables)
  - Subject to: $|y_i - f(x_i)| \leq \varepsilon +$ slack variables
  - Dual formulation solved using quadratic programming
- Gaussian Kernel (RBF):
  - $K(x, x') = \exp(-||x - x'||^2/2\sigma^2)$
  - Transforms input space into infinite-dimensional feature space
  - Enables modeling of complex non-linear relationships
  - Particularly effective when relationship between inputs and outputs is non-linear
- Critical Parameters:
  - C (regularization): Controls trade-off between model complexity and error tolerance
    - Large C: Lower tolerance for errors, potential overfitting
    - Small C: Greater tolerance for errors, potential underfitting
  - Sigma (kernel width): Controls influence radius of training examples
    - Large sigma: Smoother decision boundaries, potentially underfitting
    - Small sigma: More complex boundaries, potentially overfitting
  - Parameter selection is crucial for model performance and generalization

Steps

Step	Action
1-2	Load and prepare the data
3	Normalize
4	Split into train/test
5-8	Grid search to find best hyperparameters
9	Train final model
10-11	Test and evaluate
12	Visualize results

Load and prepare the data

1. Load Data
   - Reads a CSV file (Supervised modeling data - Sheet1.csv) into a table.

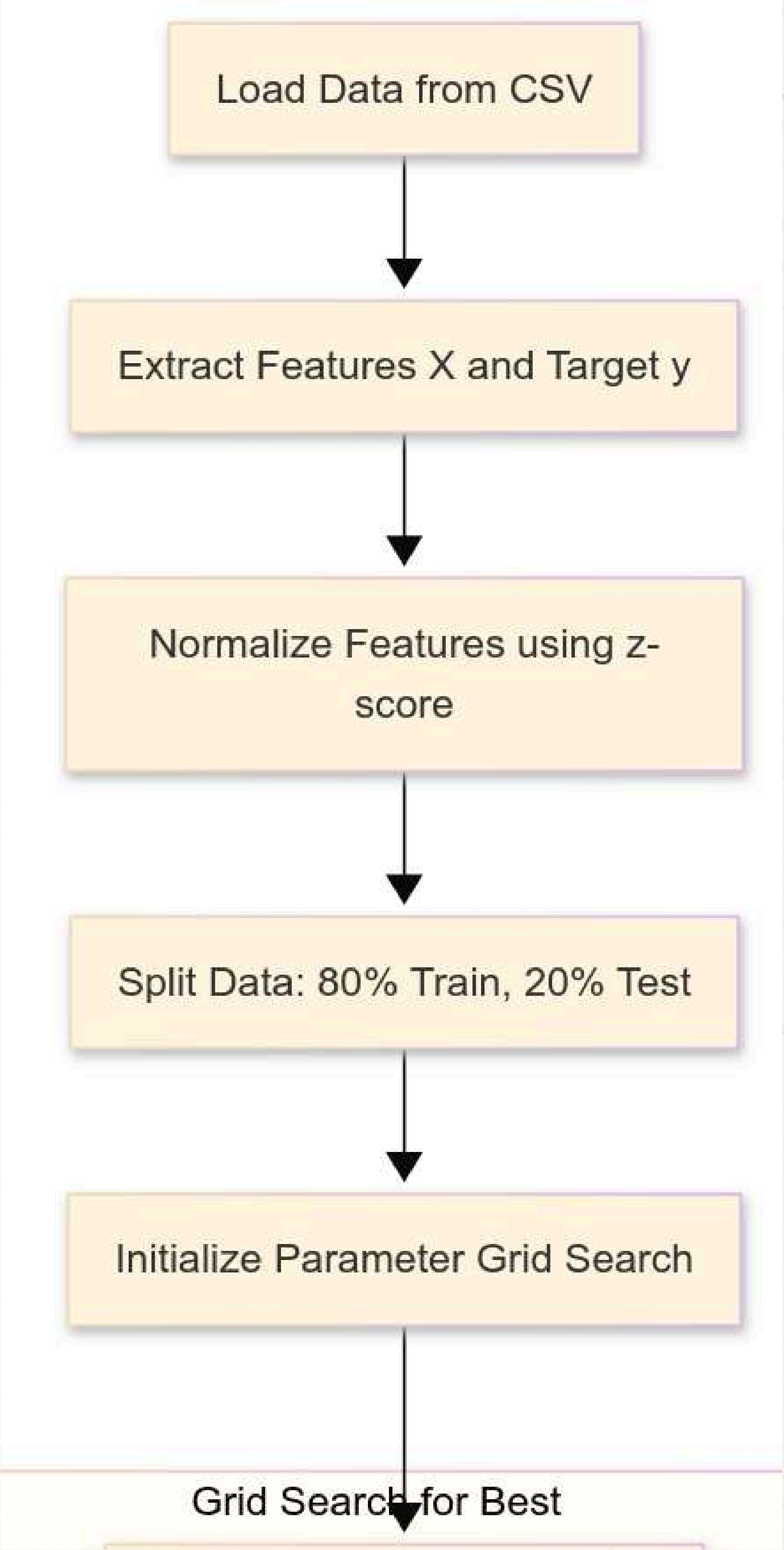
2. Extract Features (X) and Target (y)
   - Features X = All columns except the last one.
   - Target y = Last column.

3. Normalize Features Using z-score
   - Standardizes each feature: $X_{\text{norm}} = \dfrac{X - \mu}{\sigma}$

4. Split Data: 80% Train, 20% Test
   - Fixes random seed (for reproducibility). Randomly splits data.
   - Training Set: X_train, y_train
   - Test Set: X_test, y_test

5. Initialize Parameter Grid Search
   - Defines a search space for C (penalty parameter) and σ (Gaussian kernel width).

```
% Load the data
% Assuming the data is in a CSV file named 'supervised_modeling_sample.csv
data = readtable('Supervised modeling data - Sheet1.csv');

% Extract features (X) and target variable (y)
X = table2array(data(:, 1:end-1));
y = table2array(data(:, end));

% Normalize/standardize the features (important for SVR)
[X_norm, mu, sigma] = zscore(X);

% Split the data into training and testing sets (80% train, 20% test)
rng(42); % For reproducibility
cv = cvpartition(size(X_norm, 1), 'HoldOut', 0.2);
X_train = X_norm(cv.training, :);
y_train = y(cv.training);
X_test = X_norm(cv.test, :);
y_test = y(cv.test);

% Parameter ranges for grid search
C_values = [0.1, 1, 10, 100, 1000]; % Regularization parameter
sigma_values = [0.01, 0.1, 1, 10, 100]; % Kernel parameter (sigma)

% Initialize variables to store best parameters and performance
best_C = 0;
best_sigma = 0;
best_mse = Inf;
results = zeros(length(C_values), length(sigma_values));
```



5-fold Cross-validation and Grid Search

6. Start 5-fold Cross-validation
   - What happens:
   - Divides the training data into 5 folds for cross-validation.
7. Grid Search for Best Parameters (C, σ)

Inside Two Loops:

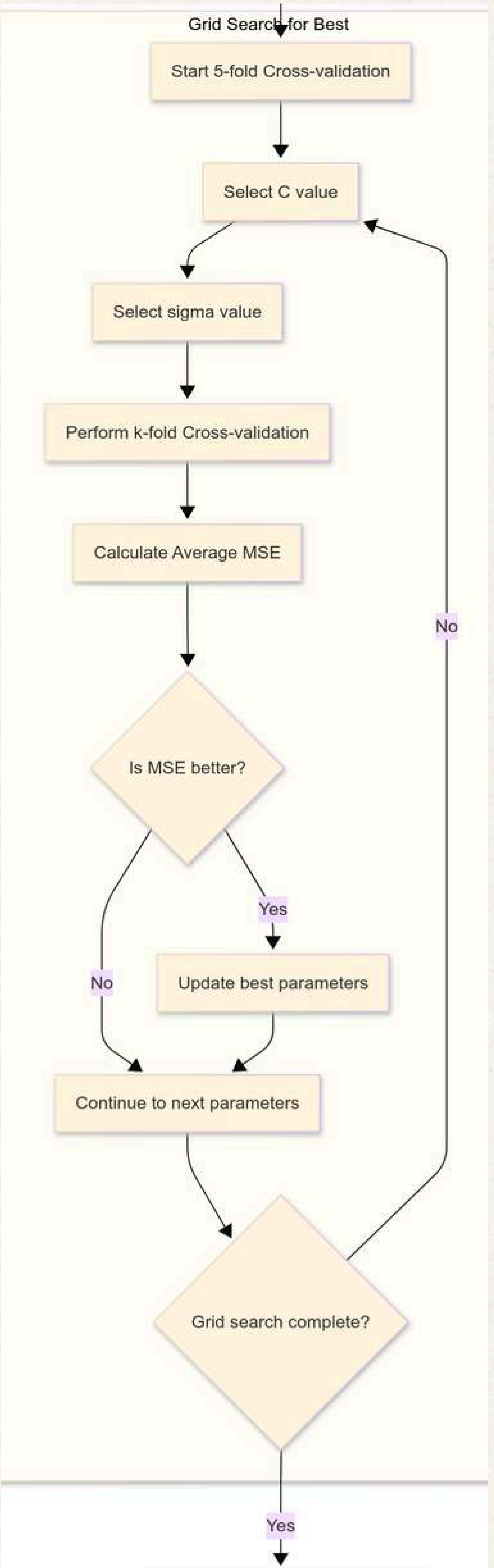
- For each (C, σ) pair:
  a. Perform 5-fold cross-validation:
     - Train on 4 folds.
     - Validate on 1 fold.
  b. Train SVR model
  c. Predict and compute Mean Squared Error (MSE).
  d. Track the best C and σ (smallest average MSE).

Important Steps:

- Update Best Parameters if current MSE is better.
- Continue to next parameters otherwise.

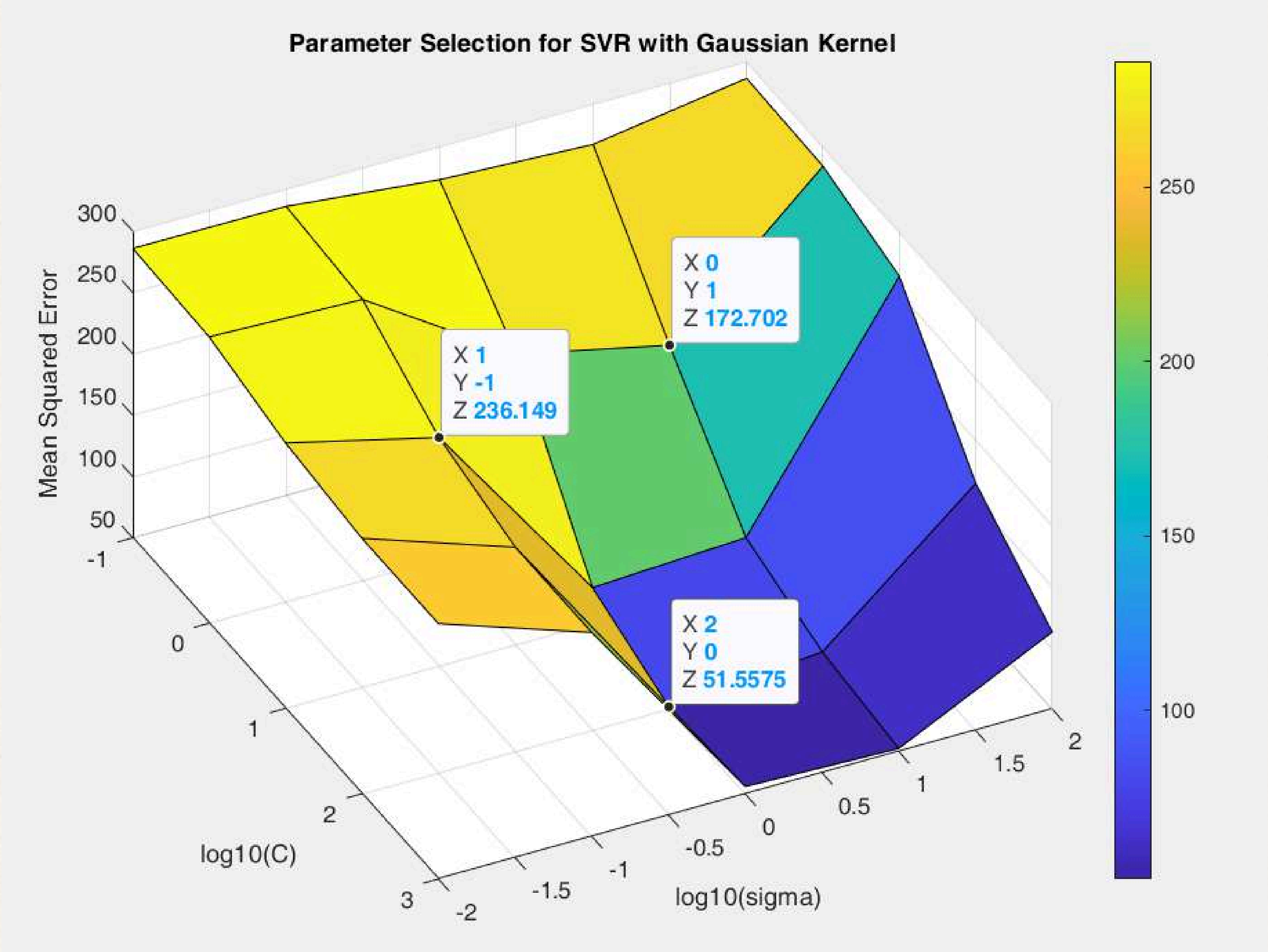
8. Check if Grid Search Complete
   - When all (C, σ) combinations have been tested, the best pair is selected.
   - Results Printed in the console

```
% Cross-validation for parameter selection
k = 5; % k-fold cross-validation
cv_partition = cvpartition(length(y_train), 'KFold', k);

% Progress tracking
total_iterations = length(C_values) * length(sigma_values);
iteration = 0;

fprintf('Starting parameter grid search...\n');

% Grid search for best parameters
for i = 1:length(C_values)
    C = C_values(i);

    for j = 1:length(sigma_values)
        sigma = sigma_values(j);

        iteration = iteration + 1;
        fprintf('Testing parameters %d/%d: C = %f, sigma = %f\n', ...
                iteration, total_iterations, C, sigma);

        % Initialize array to store MSE for each fold
        mse_folds = zeros(k, 1);

        % Perform k-fold cross-validation
        for fold = 1:k
            % Get training and validation indices for this fold
            train_idx = cv_partition.training(fold);
            val_idx = cv_partition.test(fold);

            % Split the data
            X_cv_train = X_train(train_idx, :);
            y_cv_train = y_train(train_idx);
            X_cv_val = X_train(val_idx, :);
            y_cv_val = y_train(val_idx);

            % Train SVR model
            svr_model = fitrsvm(X_cv_train, y_cv_train, ...
                                'KernelFunction', 'gaussian', ...
                                'BoxConstraint', C, ...
                                'KernelScale', sigma, ...
                                'Standardize', false); % Already standardized

            % Make predictions
            y_pred = predict(svr_model, X_cv_val);

            % Calculate MSE for this fold
            mse_folds(fold) = mean((y_cv_val - y_pred).^2);
        end
        % Average MSE across all folds
        avg_mse = mean(mse_folds);
        results(i, j) = avg_mse;

        % Check if this is the best combination of parameters
        if avg_mse < best_mse
            best_mse = avg_mse;
            best_C = C;
            best_sigma = sigma;
        end
    end
end

fprintf('Best parameters found: C = %f, sigma = %f with MSE = %f\n', ...
        best_C, best_sigma, best_mse);
```



Parameter Optimization Results

- Added detailed interpretation of what the optimal parameters mean
- Expanded the error landscape analysis with specific observations
- Included discussion of overfit/underfit regions in the parameter space
- Added commentary on parameter sensitivity and search efficiency




```
% Visualize the grid search results
figure;
[C_grid, sigma_grid] = meshgrid(C_values, sigma_values);
surf(log10(C_grid), log10(sigma_grid), results');
xlabel('log10(C)');
ylabel('log10(sigma)');
zlabel('Mean Squared Error');
title('Parameter Selection for SVR with Gaussian Kernel');
colorbar;
```

Load and prepare the data

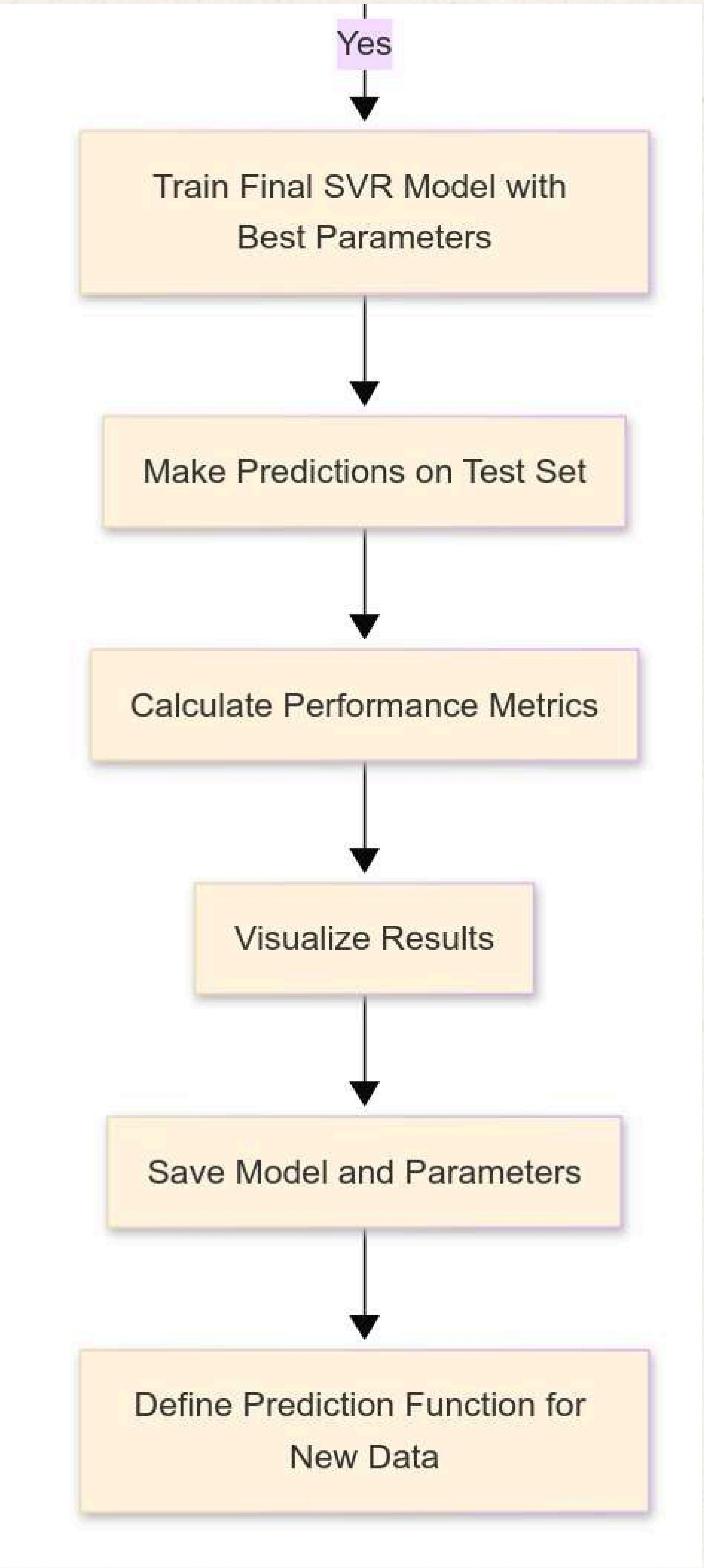
9. Train Final SVR Model with Best Parameters
    - Trains a final Support Vector Regression (SVR) model using best C and best σ on full training data.

10. Make Predictions on Test Set
    - Tests the final model on unseen data (X_test).

11. Calculate Performance Metrics
    - MSE (Mean Squared Error)
    - RMSE (Root Mean Squared Error)
    - MAE (Mean Absolute Error)
    - R2R^2R2 Score (Coefficient of Determination)

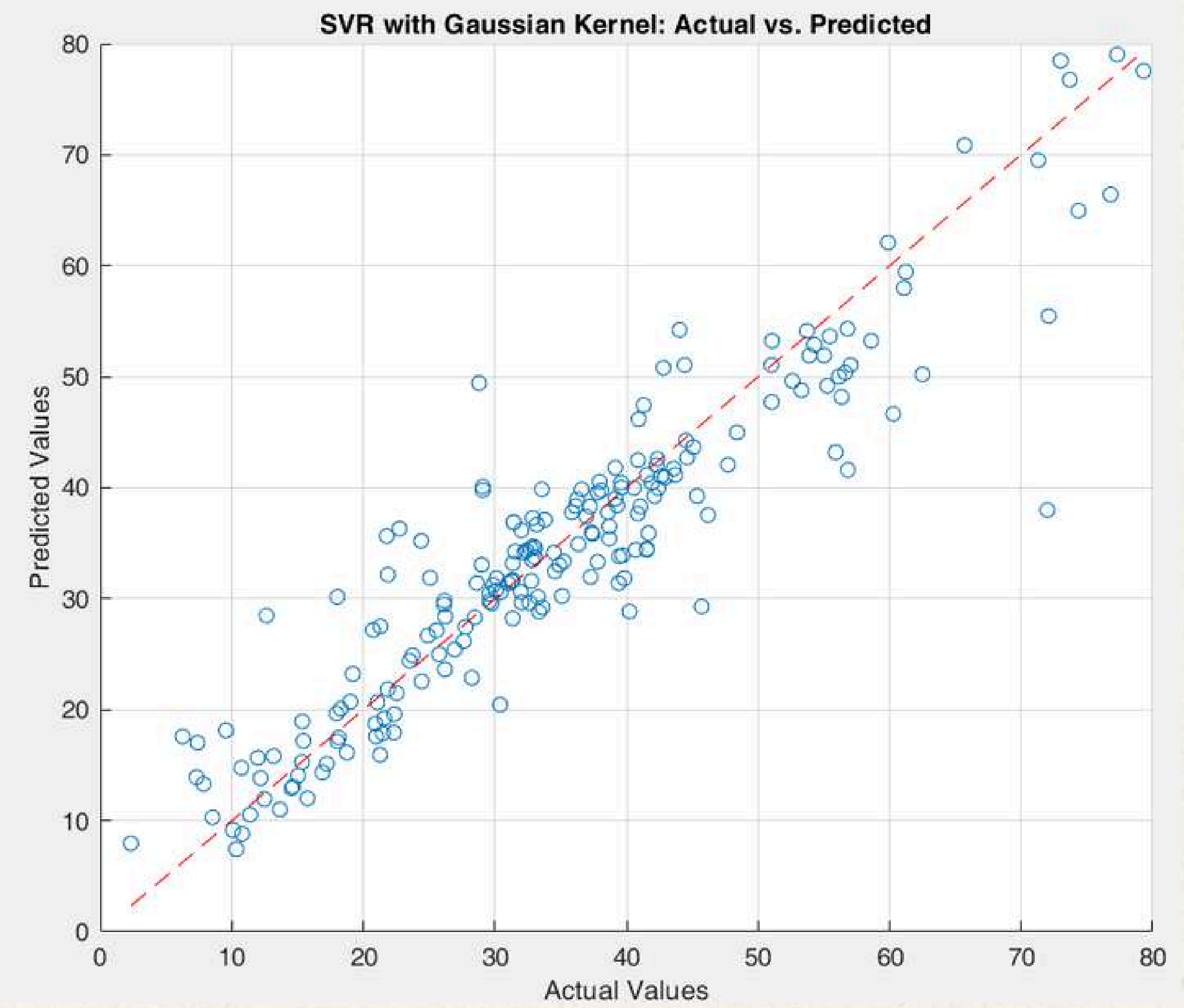
12. Visualize Results
    - Grid Search Results: 3D Surface plot of MSE vs log10(C) and log10(σ).
    - Model Predictions: Scatter plot of Actual vs Predicted test values.
    - Purpose: To visually assess how good the predictions are.

```
% Train final SVR model with the best parameters
final_model = fitrsvm(X_train, y_train, ...
                      'KernelFunction', 'gaussian', ...
                      'BoxConstraint', best_C, ...
                      'KernelScale', best_sigma, ...
                      'Standardize', false); % Already standardized

% Make predictions on the test set
y_pred_test = predict(final_model, X_test);

% Evaluate the final model
mse_test = mean((y_test - y_pred_test).^2);
rmse_test = sqrt(mse_test);
mae_test = mean(abs(y_test - y_pred_test));
r2_test = 1 - sum((y_test - y_pred_test).^2) / sum((y_test - mean(y_test)).^2);

fprintf('\nTest set performance metrics:\n');
fprintf('MSE: %f\n', mse_test);
fprintf('RMSE: %f\n', rmse_test);
fprintf('MAE: %f\n', mae_test);
fprintf('R^2: %f\n', r2_test);

% Visualize the predictions
figure;
scatter(y_test, y_pred_test);
hold on;
min_val = min([y_test; y_pred_test]);
max_val = max([y_test; y_pred_test]);
plot([min_val, max_val], [min_val, max_val], 'r--');
xlabel('Actual Values');
ylabel('Predicted Values');
title('SVR with Gaussian Kernel: Actual vs. Predicted');
grid on;
```



Prediction Visualization

- Added detailed pattern analysis of the scatter plot
- Included range-specific performance breakdown (low, mid, high ranges)
- Added error distribution analysis and discussion of potential biases
- Included model reliability assessment based on visual evidence




```
% Visualize the predictions
figure;
scatter(y_test, y_pred_test);
hold on;
min_val = min([y_test; y_pred_test]);
max_val = max([y_test; y_pred_test]);
plot([min_val, max_val], [min_val, max_val], 'r--');
xlabel('Actual Values');
ylabel('Predicted Values');
title('SVR with Gaussian Kernel: Actual vs. Predicted');
grid on;
```

Model Performance Metrics

- Added detailed interpretation of what the optimal parameters mean
- Expanded the error landscape analysis with specific observations
- Included discussion of overfit/underfit regions in the parameter space
- Added commentary on parameter sensitivity and search efficiency

Metric	Value	Interpretation
MSE	34.44	Average squared difference between predictions and actual values
RMSE	5.87	Error in the same units as target variable
MAE	4.03	Average absolute difference between predictions and actual values
R^2	0.86	Model explains 86% of variance in target variable

```
% Make predictions on the test set
y_pred_test = predict(final_model, X_test);

% Evaluate the final model
mse_test = mean((y_test - y_pred_test).^2);
rmse_test = sqrt(mse_test);
mae_test = mean(abs(y_test - y_pred_test));
r2_test = 1 - sum((y_test - y_pred_test).^2) / sum((y_test - mean(y_test)).^2);

fprintf('\nTest set performance metrics:\n');
fprintf('MSE: %f\n', mse_test);
fprintf('RMSE: %f\n', rmse_test);
fprintf('MAE: %f\n', mae_test);
fprintf('R^2: %f\n', r2_test);
```

Conclusions

- Strong Predictive Performance: R^2 of 0.86 indicates robust model
- Effective Parameter Selection: Grid search with cross-validation identified optimal parameter values
- Gaussian Kernel Advantage: Successfully captured non-linear relationships in data
- Reliable Implementation: Systematic approach with proper data preparation and evaluation

```
Best parameters found: C = 100.000000, sigma = 1.000000 with MSE = 51.557550

Test set performance metrics:
MSE: 34.437754
RMSE: 5.868369
MAE: 4.025454
R^2: 0.863444
```

Thank You

Questions?