```
% --------------------
% Load and Prepare Dataset
% --------------------
% Load the dataset with preserved column names
data = readtable('/MATLAB Drive/international-airline-passengers.csv',
'VariableNamingRule', 'preserve');
```

```
Warning: Table variable names were truncated to the length namelengthmax. The original names are
saved in the VariableDescriptions property.
```

```

% Ensure variable names are valid and within the allowed length
data.Properties.VariableNames =
matlab.lang.makeValidName(data.Properties.VariableNames);

% Rename columns explicitly for simplicity and clarity
data.Properties.VariableNames = {'Month', 'Passengers'};

% Remove rows with missing values
data = rmmissing(data);

% Convert 'Month' to datetime format
data.Month = datetime(data.Month, 'InputFormat', 'yyyy-MM');

% Apply a moving average to smooth passenger data
data.Passengers = movmean(data.Passengers, 3);

% Add a logarithmic transformation column
data.LogPassengers = log(data.Passengers);

% --------------------
% Split Data into Training and Testing Sets
% --------------------
% Split data into 80% training and 20% testing
splitIdx = floor(0.8 * height(data));
trainData = data(1:splitIdx, :);
testData = data(splitIdx+1:end, :);

% Feature Engineering: Lag features and seasonality
trainData.Lag1 = [NaN; trainData.Passengers(1:end-1)];
trainData.Lag2 = [NaN; NaN; trainData.Passengers(1:end-2)];
trainData.MonthSin = sin(2 * pi * month(trainData.Month) / 12);
trainData.MonthCos = cos(2 * pi * month(trainData.Month) / 12);
trainData = rmmissing(trainData);

% Repeat for testing data
testData.Lag1 = [NaN; testData.Passengers(1:end-1)];
testData.Lag2 = [NaN; NaN; testData.Passengers(1:end-2)];
testData.MonthSin = sin(2 * pi * month(testData.Month) / 12);
testData.MonthCos = cos(2 * pi * month(testData.Month) / 12);
testData = rmmissing(testData);
```

```
% Extract features and target variables
X_train = [trainData.Lag1, trainData.Lag2, trainData.MonthSin,
trainData.MonthCos];
y_train = trainData.Passengers;
X_test = [testData.Lag1, testData.Lag2, testData.MonthSin,
testData.MonthCos];
y_test = testData.Passengers;

% Normalize features for training and testing
X_train_norm = (X_train - min(X_train)) ./ (max(X_train) - min(X_train));
X_test_norm = (X_test - min(X_train)) ./ (max(X_train) - min(X_train));

% -------------------
% Train and Evaluate Models
% -------------------

% Linear Regression
lm = fitlm(X_train_norm, y_train);
y_pred_lm = predict(lm, X_test_norm);
disp('Linear Regression Results:');
```

```
Linear Regression Results:
```

```
disp(lm);
```

```
Linear regression model:
    y ~ 1 + x1 + x2 + x3 + x4

Estimated Coefficients:
                   Estimate      SE        tStat        pValue
                   ________    ______    _______    __________

    (Intercept)     107.18     3.1756     33.751    3.3819e-59
    x1               538.1      28.39     18.954    3.8799e-36
    x2             -200.47     28.723    -6.9795    2.5045e-10
    x3               17.89     3.1648     5.6528    1.3009e-07
    x4             -3.9563     3.8983    -1.0149       0.31243


Number of observations: 113, Error degrees of freedom: 108
Root Mean Squared Error: 10.9
R-squared: 0.985,  Adjusted R-Squared: 0.985
F-statistic vs. constant model: 1.82e+03, p-value = 4.61e-98
```

```
% Support Vector Regression (SVR)
svmModel = fitrsvm(X_train_norm, y_train, ...
    'KernelFunction', 'gaussian', ...
    'BoxConstraint', 10, ...
    'Epsilon', 0.1);
y_pred_svr = predict(svmModel, X_test_norm);

% Random Forest Regressor
numTrees = 200; % Number of trees
```

```
rfModel = TreeBagger(numTrees, X_train_norm, y_train, ...
    'Method', 'regression', ...
    'MaxNumSplits', 10, ...
    'MinLeafSize', 5);
y_pred_rf = str2double(predict(rfModel, X_test_norm));

% --------------------
% Model Evaluation (R² and RMSE)
% --------------------
r2_lm = 1 - sum((y_test - y_pred_lm).^2) / sum((y_test - mean(y_test)).^2);
r2_svr = 1 - sum((y_test - y_pred_svr).^2) / sum((y_test - mean(y_test)).^2);
r2_rf = 1 - sum((y_test - y_pred_rf).^2) / sum((y_test - mean(y_test)).^2);

rmse_lm = sqrt(mean((y_test - y_pred_lm).^2));
rmse_svr = sqrt(mean((y_test - y_pred_svr).^2));
rmse_rf = sqrt(mean((y_test - y_pred_rf).^2));

% Display results
fprintf('Linear Regression R^2: %.2f, RMSE: %.2f\n', r2_lm, rmse_lm);
```

```
Linear Regression R^2: 0.91, RMSE: 21.60
```

```
fprintf('SVR R^2: %.2f, RMSE: %.2f\n', r2_svr, rmse_svr);
```

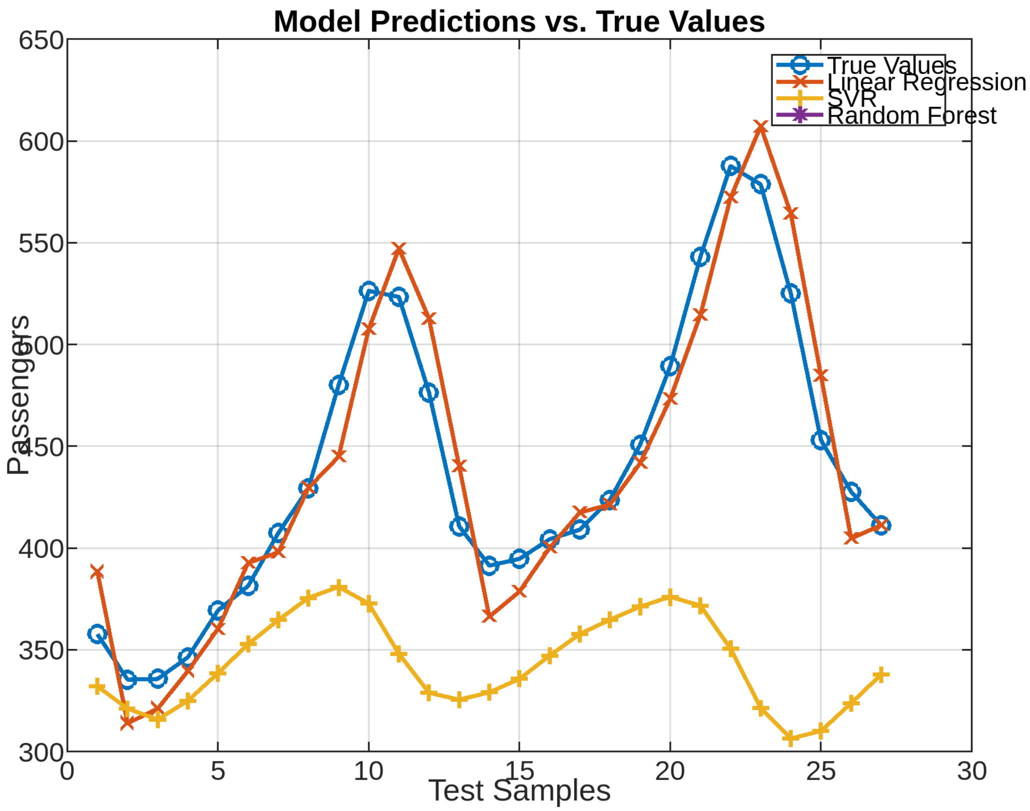
```
SVR R^2: -1.77, RMSE: 117.60
```

```
fprintf('Random Forest R^2: %.2f, RMSE: %.2f\n', r2_rf, rmse_rf);
```

```
Random Forest R^2: NaN, RMSE: NaN
```

```

% --------------------
% Plot Results
% --------------------
figure;
plot(y_test, 'o-', 'DisplayName', 'True Values', 'LineWidth', 1.5);
hold on;
plot(y_pred_lm, 'x-', 'DisplayName', 'Linear Regression', 'LineWidth', 1.5);
plot(y_pred_svr, '+-', 'DisplayName', 'SVR', 'LineWidth', 1.5);
plot(y_pred_rf, '*-', 'DisplayName', 'Random Forest', 'LineWidth', 1.5);
legend('show', 'Location', 'best');
xlabel('Test Samples');
ylabel('Passengers');
title('Model Predictions vs. True Values');
grid on;
hold off;
```


Model Predictions vs. True Values
True Values
Linear Regression
SVR
Random Forest
Passengers
Test Samples
650
600
550
500
450
400
350
300
0
5
10
15
20
25
30