```
% Generate synthetic data
N = 20; % or 100
R = 200; %No. of realisations
iterations = 1000;
alpha = 0.01; % Learning rate

% Initialize arrays to store beta coefficients
beta_ols = [];
beta_lms = [];
beta_lts = [];

for i = 1:R
    % Generate synthetic data
    X1 = randn(N, 1);
    X2 = randn(N, 1);
    epsilon = randn(N, 1);
    y = 3*X1 + 5*X2 + epsilon;
    X = [X1, X2];

    % Estimate parameters using different methods
    beta_ols = [beta_ols, gradient_descent_ols_with_bias(X, y, alpha, iterations)];
    beta_lms = [beta_lms, gradient_descent_lms_with_bias(X, y, alpha, iterations)];
    beta_lts = [beta_lts, gradient_descent_lts_with_bias(X, y, alpha, iterations)];
end

% Compute metrics
compute_and_display_metrics(X, y, beta_ols, beta_lms, beta_lts);
```

```
Metrics for OLS:
    "MSE: 9.72119"    "MSE: 9.89736"    "MSE: 5.38768"    "MSE: 6.81587"    "MSE: 4.99708"    "MSE: 5.81041"    "MSE

    "RB: -1.45"    "RB: -1.4752"    "RB: -0.74581"    "RB: -1.0284"    "RB: -0.62346"    "RB: -0.83652"    "RB: -1.2
    "RB: -3.45"    "RB: -3.4752"    "RB: -2.7458"     "RB: -3.0284"    "RB: -2.6235"     "RB: -2.8365"     "RB: -3.2

    "MAD: 2.45"    "MAD: 2.4752"    "MAD: 1.7458"    "MAD: 2.0284"    "MAD: 1.6235"    "MAD: 1.8365"    "MAD: 2.2628

Metrics for LMS:
    "MSE: 23.1625"    "MSE: 23.5087"    "MSE: 22.9763"    "MSE: 22.6244"    "MSE: 23.3276"    "MSE: 23.0365"    "MSE

    "RB: -2.9643"    "RB: -2.9861"    "RB: -2.9433"    "RB: -2.9164"    "RB: -2.9379"    "RB: -2.9357"    "RB: -2.97
    "RB: -4.9643"    "RB: -4.9861"    "RB: -4.9433"    "RB: -4.9164"    "RB: -4.9379"    "RB: -4.9357"    "RB: -4.97

    "MAD: 3.9643"    "MAD: 3.9861"    "MAD: 3.9433"    "MAD: 3.9164"    "MAD: 3.9379"    "MAD: 3.9357"    "MAD: 3.97

Metrics for LTS:
    "MSE: 21.4162"    "MSE: 20.3018"    "MSE: 19.8992"    "MSE: 19.8979"    "MSE: 18.7858"    "MSE: 16.7653"    "MSE

    "RB: -2.7957"    "RB: -2.7299"    "RB: -2.4897"    "RB: -2.5732"    "RB: -2.4949"    "RB: -2.3495"    "RB: -2.83
    "RB: -4.7957"    "RB: -4.7299"    "RB: -4.4897"    "RB: -4.5732"    "RB: -4.4949"    "RB: -4.3495"    "RB: -4.83

    "MAD: 3.7957"    "MAD: 3.7299"    "MAD: 3.4897"    "MAD: 3.5732"    "MAD: 3.4949"    "MAD: 3.3495"    "MAD: 3.83
```

```
% Import and prepare medical insurance data
```

```
medical_insurance = medicalinsurance;

% Convert categorical variables to numerical
[~, ~, sex] = unique(medical_insurance.sex);
[~, ~, smoker] = unique(medical_insurance.smoker);
[~, ~, region] = unique(medical_insurance.region);

data = [medical_insurance.age, medical_insurance.bmi, medical_insurance.children, sex, smoker,
target = medical_insurance.charges;

% Split data into training and test sets
n = size(data, 1);
idx = randperm(n);
train_idx = idx(1:round(0.8*n));
test_idx = idx(round(0.8*n)+1:end);

X_train = data(train_idx, :);
y_train = target(train_idx);
X_test = data(test_idx, :);
y_test = target(test_idx);

% Parameter estimation using different methods
alpha = 1e-9;
iterations = 100000;
beta_ols = gradient_descent_ols_with_bias(X_train, y_train, alpha, iterations);
beta_lms = gradient_descent_lms_with_bias(X_train, y_train, alpha, iterations);
beta_lts = gradient_descent_lts_with_bias(X_train, y_train, alpha, iterations);

% Compute predictions and MSE for test data
X_test_with_bias = [ones(size(X_test, 1), 1), X_test];
y_pred_ols = X_test_with_bias * beta_ols;
y_pred_lms = X_test_with_bias * beta_lms;
y_pred_lts = X_test_with_bias * beta_lts;

MSE_ols = mean((y_test - y_pred_ols).^2);
MSE_lms = mean((y_test - y_pred_lms).^2);
MSE_lts = mean((y_test - y_pred_lts).^2);

%Display the MSE, divide by 10^8 so as to normalise.
disp("MSE for OLS: " + MSE_ols/10^8);
```

```
MSE for OLS: 1.9023
```

```
disp("MSE for LMS: " + MSE_lms/10^8);
```

```
MSE for LMS: 3.1156
```

```
disp("MSE for LTS: " + MSE_lts/10^8);
```

```
MSE for LTS: 2.6158
```

```

```

We can see that OLS gives the least MSE

```
% Function Definitions
```

```
function beta = gradient_descent_ols_with_bias(X, y, alpha, iterations)
    [n, p] = size(X);
    X = [ones(n, 1), X]; % Include bias term
    beta = zeros(p + 1, 1);
    lambda = 1;
    for iter = 1:iterations
        residuals = y - X * beta;
        gradient = -2 * X' * residuals / n + 2 * lambda * beta;
        beta = beta - alpha * gradient;
    end
end

function beta = gradient_descent_lms_with_bias(X, y, alpha, iterations)
    [n, p] = size(X);
    X = [ones(n, 1), X]; % Include bias term
    beta = zeros(p + 1, 1);
    lambda = 2;
    for iter = 1:iterations
        residuals = y - X * beta;
        sorted_residuals = sort(residuals.^2);
        median_residual = sorted_residuals(floor(n/2) + 1);
        median_residual_idx = find(residuals.^2 == median_residual);
        gradient = -2 * X(median_residual_idx, :)' * residuals(median_residual_idx) / n + 2 * l
        beta = beta - alpha * gradient;
    end
end

function beta = gradient_descent_lts_with_bias(X, y, alpha, iterations)
    [n, p] = size(X);
    X = [ones(n, 1), X]; % Include bias term
    q = floor(n/2) + 1;
    beta = zeros(p + 1, 1);
    lambda = 2;
    for iter = 1:iterations
        residuals = y - X * beta;
        sorted_residuals = sort(residuals.^2);
        trimmed_residuals = sorted_residuals(1:q);
        trimmed_residuals_idx = ismember(residuals.^2, trimmed_residuals);
        gradient = -2 * sum(X(trimmed_residuals_idx, :) .* residuals(trimmed_residuals_idx), 1)
        beta = beta - alpha * gradient;
    end
```

```
end

function compute_and_display_metrics(X, y, beta_ols, beta_lms, beta_lts)
    X_with_bias = [ones(size(X, 1), 1), X];
    y_pred_ols = X_with_bias * beta_ols;
    y_pred_lms = X_with_bias * beta_lms;
    y_pred_lts = X_with_bias * beta_lts;

    MSE_ols = mean((y - y_pred_ols).^2, 1);
    MSE_lms = mean((y - y_pred_lms).^2, 1);
    MSE_lts = mean((y - y_pred_lts).^2, 1);

    RB_ols = median(beta_ols(2:3, :), 1) - [3; 5];
    RB_lms = median(beta_lms(2:3, :), 1) - [3; 5];
    RB_lts = median(beta_lts(2:3, :), 1) - [3; 5];

    MAD_ols = median(abs(beta_ols(2:3, :) - [3; 5]), 1);
    MAD_lms = median(abs(beta_lms(2:3, :) - [3; 5]), 1);
    MAD_lts = median(abs(beta_lts(2:3, :) - [3; 5]), 1);

    disp("Metrics for OLS:");
    disp("MSE: " + MSE_ols);
    disp("RB: " + RB_ols);
    disp("MAD: " + MAD_ols);
    disp("Metrics for LMS:");
    disp("MSE: " + MSE_lms);
    disp("RB: " + RB_lms);
    disp("MAD: " + MAD_lms);
    disp("Metrics for LTS:");
    disp("MSE: " + MSE_lts);
    disp("RB: " + RB_lts);
    disp("MAD: " + MAD_lts);
end
```