DS-203

ASSIGNMENT-11

- 22B1207- Anshu Arora
- 22B1219- Abhineet Agarwal
- 22B3958- Garima Gopalani

THE USUAL QUESTIONS FIRST:

What kind of EDA will you do on the data to get an overall understanding? How good is the data? Are there any parameters that are bad, in terms of data not being available? What to do with such columns? Are there other columns that are not very good but which can be 'managed'? If they can be 'managed', how? Is there a need to standardize / normalize the data? Is there a need to apply any kind of data transformation to some of the parameters?

There was a lot of missing data, we first filtered that out by:

- Making the non numeric data as 0 .
- If more than 50% of the data was 0 we dropped the column.
- If the data was extensively fluctuating, we dropped those columns as well by seeing the ratio of the std deviation and mean of the data.
- Now, for the remaining columns that had some missing data, we linearly interpolated it.
- We then saw the correlation of the columns with each other and dropped the ones with high correlation.

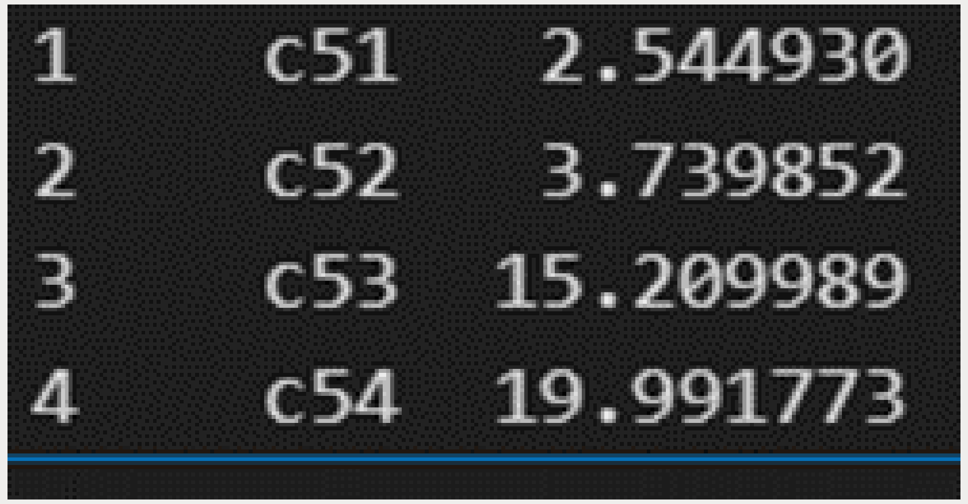
Will it be necessary to create ML models for each one of c51, c52, c53, c54? How to decide?

This would not be necessary if columns c51, c52 and c53 have a high level of correlation.

There are many columns in the data set – a fertile ground for conditions of multicollinearity. Should all these columns necessarily be used while training ML models? How to make this decision?

We used a factor called VIF. It detects multicollinearity in regression analysis, that is when some predictor variables are highly correlated, it causes redundancy and issues in the analysis.

A value greater than 10 depicts high levels of collinearity. Hence, it is not necessary to make models for c53 and c54.



```
1    c51   2.544930
2    c52   3.739852
3    c53  15.209989
4    c54  19.991773
```

THE STEPS

DATA CLEANING

- Handling non-numeric values by replacing them with zeros.
- Filtering columns based on the percentage of zero values.
- Normalizing the data for each column and dropping columns based on mean deviation.
- Interpolating zero values and identifying outliers using Z-scores.
- Addressing outliers and handling missing values by interpolation.

IDENTIFYING AND ADDRESSING HIGH CORRELATION

Calculating the correlation matrix and grouping columns based on correlation values to remove highly correlated columns.

THE STEPS

SPLITTING DATA

Dividing data into separate DataFrames for different purposes, such as separating vibration columns and specific energy columns.

VARIANCE INFLATION FACTOR (VIF) CALCULATION

- Calculating VIF for the selected predictors to detect multicollinearity.
- Removing columns with high VIF from the dataset.

LINEAR REGRESSION MODELING

Building a linear regression model using Ordinary Least Squares (OLS) and Statsmodels library.

THE STEPS

STEPWISE FEATURE SELECTION

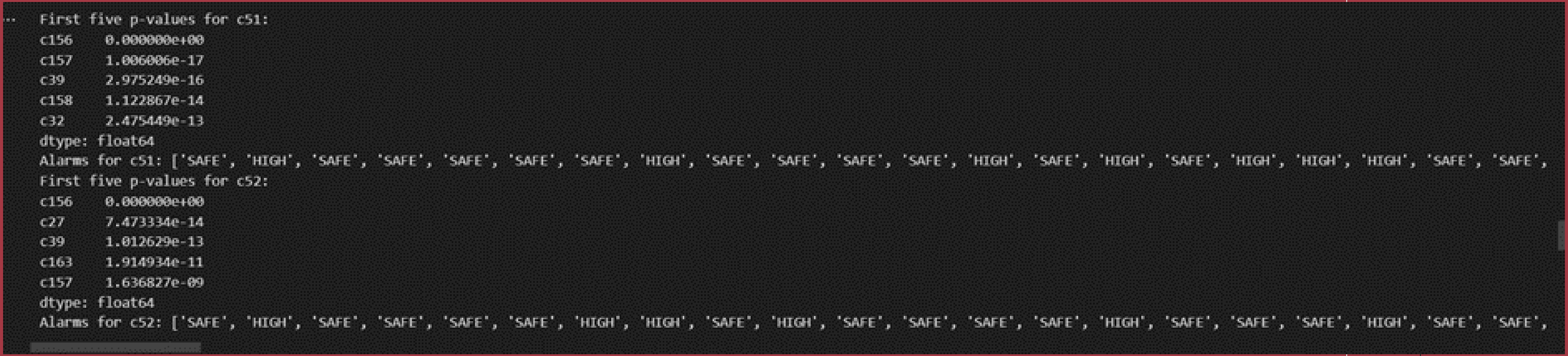
Implementing a stepwise method to select features by dropping variables with high p-values until all predictors have p-values below 0.05.

MODEL EVALUATION AND VISUALIZATION

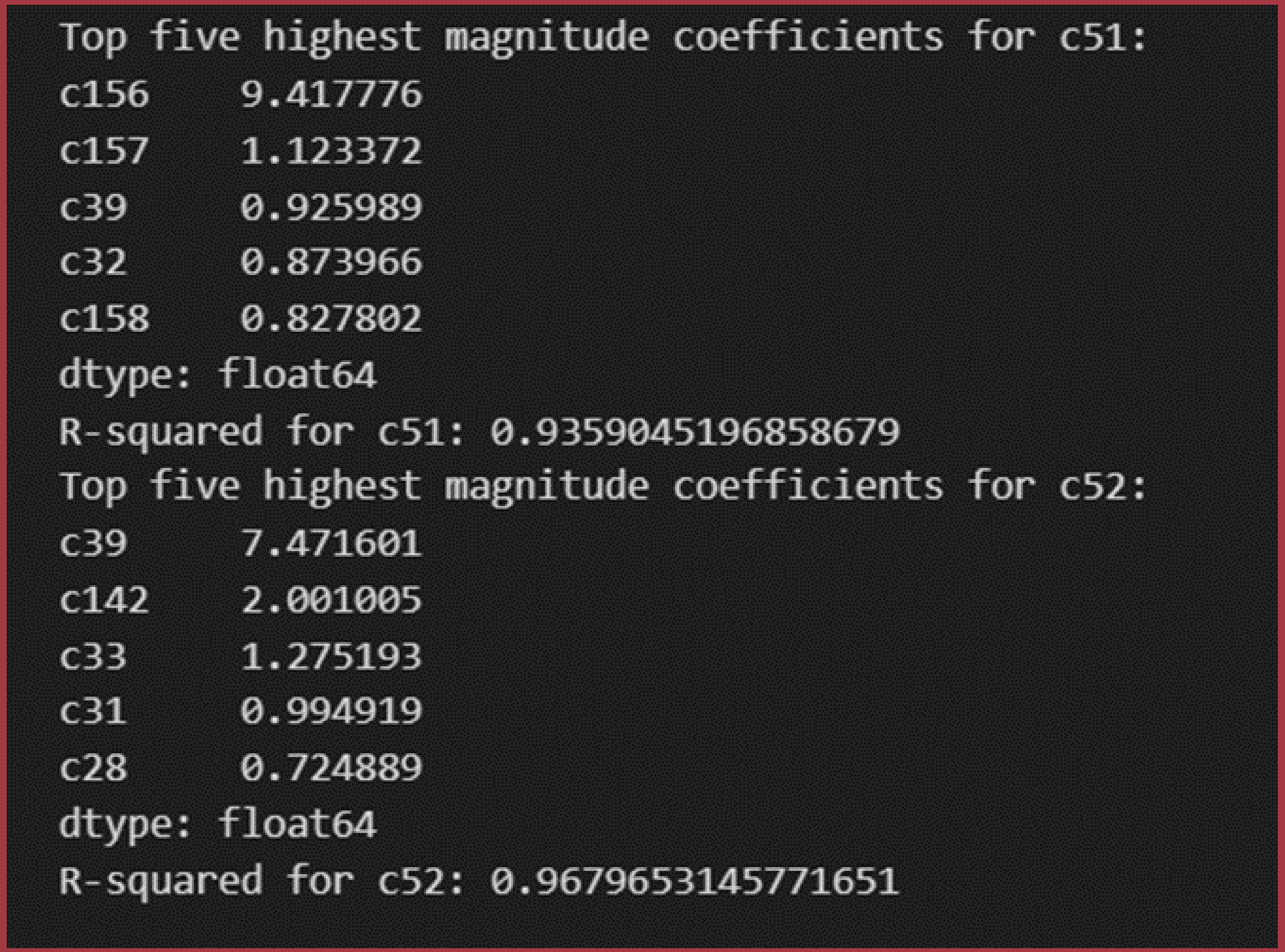
Visualizing the original and predicted values using scatter plots.

RESULTS

The below snippets are of c51 and c52 based on the controllable parameters that have been filtered out for high correlation.

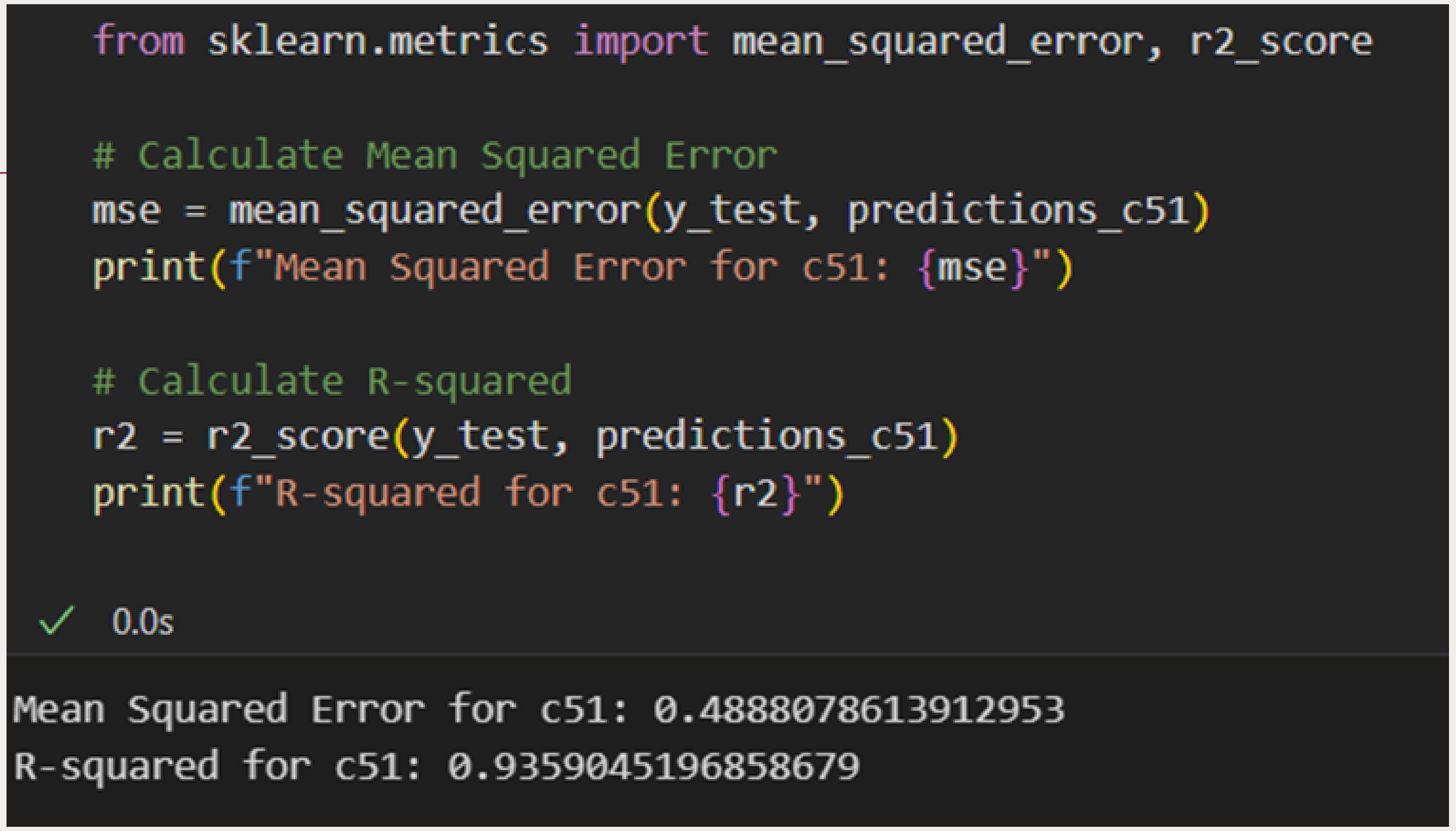


```
First five p-values for c51:
c156    0.000000e+00
c157    1.006006e-17
c39     2.975249e-16
c158    1.122867e-14
c32     2.475449e-13
dtype: float64
Alarms for c51: ['SAFE', 'HIGH', 'SAFE', 'SAFE', 'SAFE', 'SAFE', 'SAFE', 'HIGH', 'SAFE', 'SAFE', 'SAFE', 'SAFE', 'HIGH', 'SAFE', 'HIGH', 'SAFE', 'HIGH', 'HIGH', 'HIGH', 'SAFE', 'SAFE',
First five p-values for c52:
c156    0.000000e+00
c27     7.473334e-14
c39     1.012629e-13
c163    1.914934e-11
c157    1.636827e-09
dtype: float64
Alarms for c52: ['SAFE', 'HIGH', 'SAFE', 'SAFE', 'SAFE', 'SAFE', 'HIGH', 'HIGH', 'SAFE', 'HIGH', 'SAFE', 'SAFE', 'SAFE', 'SAFE', 'HIGH', 'SAFE', 'SAFE', 'SAFE', 'HIGH', 'SAFE', 'SAFE',
```



```
Top five highest magnitude coefficients for c51:
c156    9.417776
c157    1.123372
c39     0.925989
c32     0.873966
c158    0.827802
dtype: float64
R-squared for c51: 0.9359045196858679
Top five highest magnitude coefficients for c52:
c39     7.471601
c142    2.001005
c33     1.275193
c31     0.994919
c28     0.724889
dtype: float64
R-squared for c52: 0.9679653145771651
```

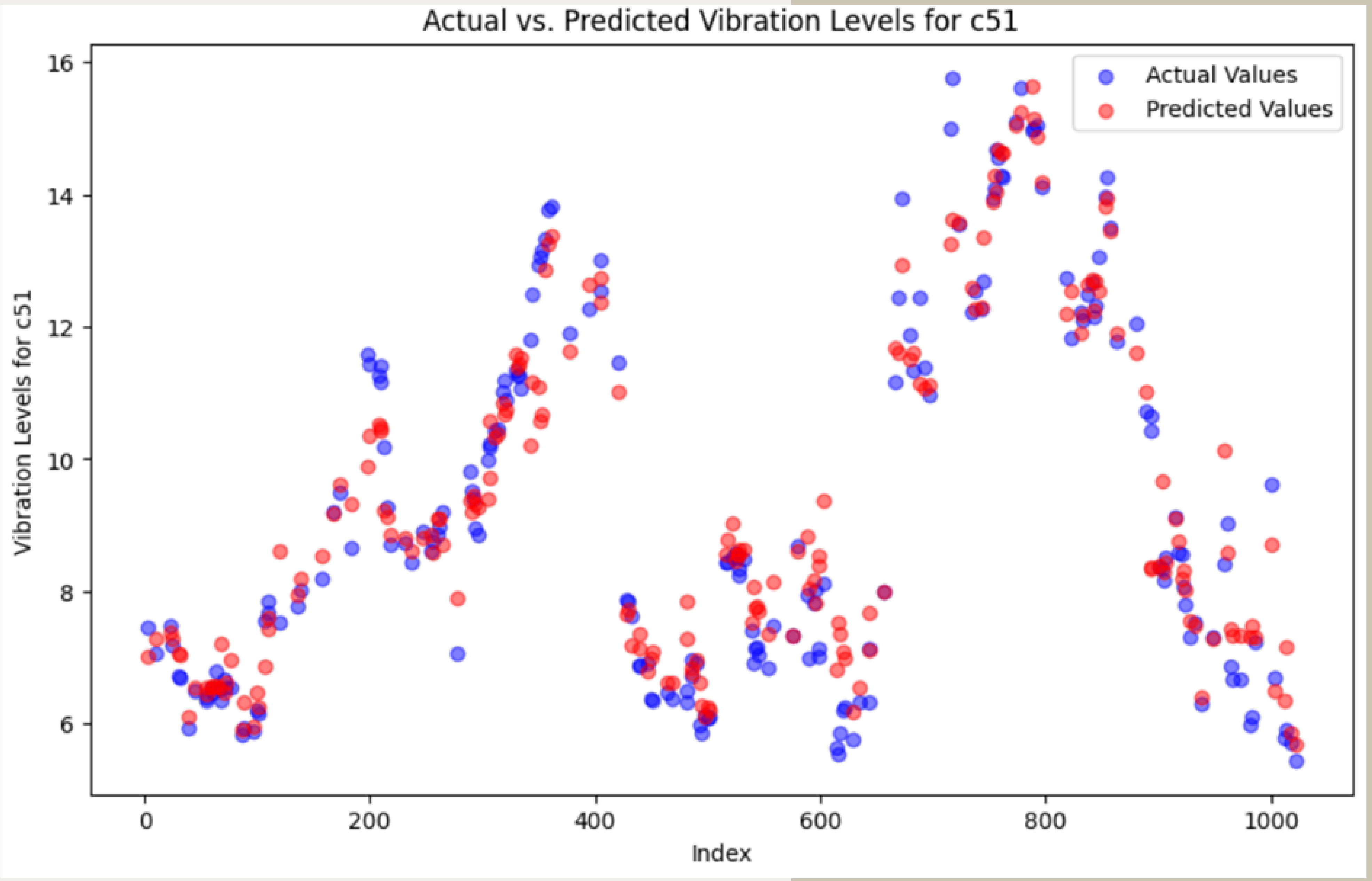
Here are the most important features for c51 and c52 (a change in them will give greatest change in the operating parameter). The output for alarm generation is also provided.

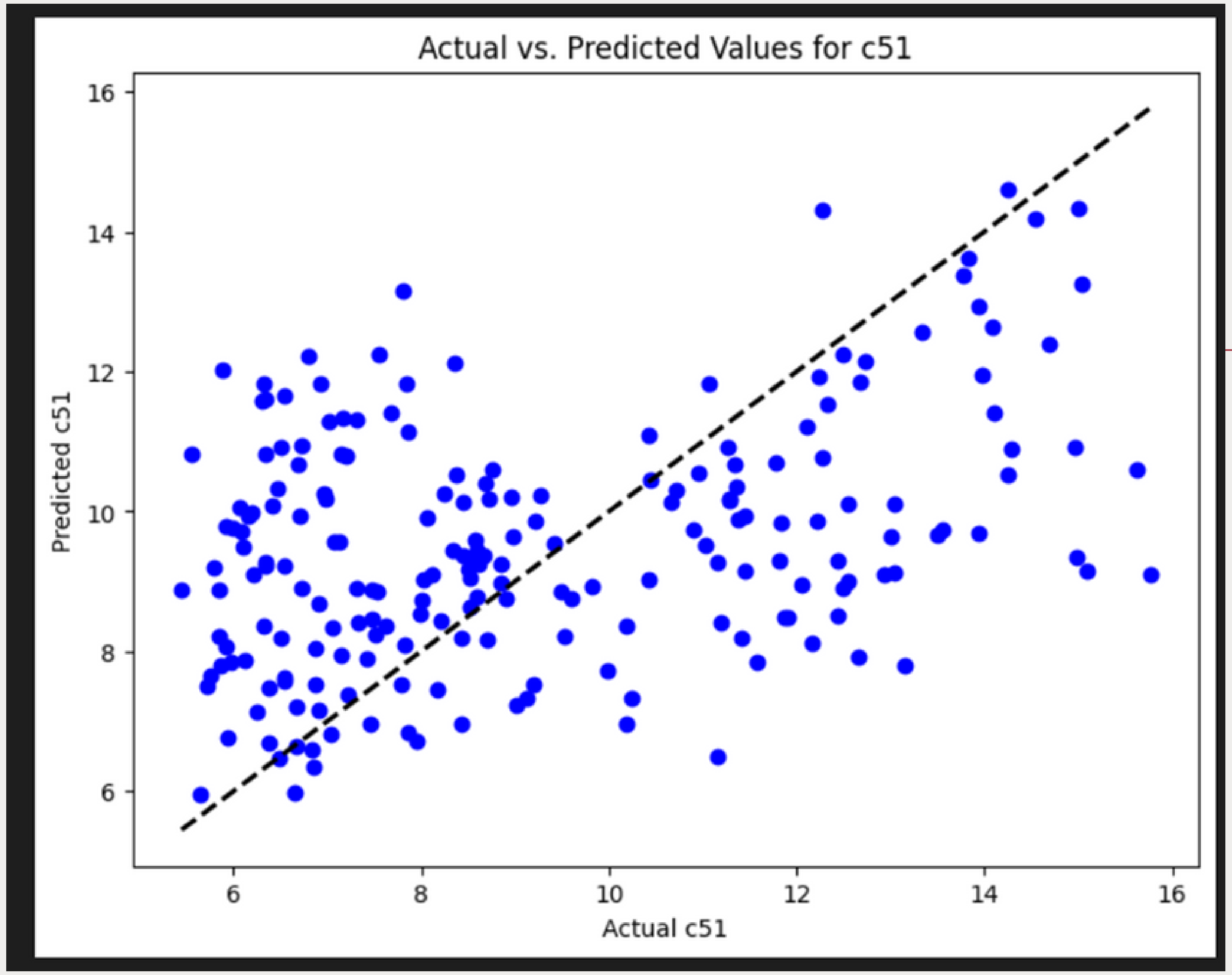


```
from sklearn.metrics import mean_squared_error, r2_score

# Calculate Mean Squared Error
mse = mean_squared_error(y_test, predictions_c51)
print(f"Mean Squared Error for c51: {mse}")

# Calculate R-squared
r2 = r2_score(y_test, predictions_c51)
print(f"R-squared for c51: {r2}")
```

✓ 0.0s

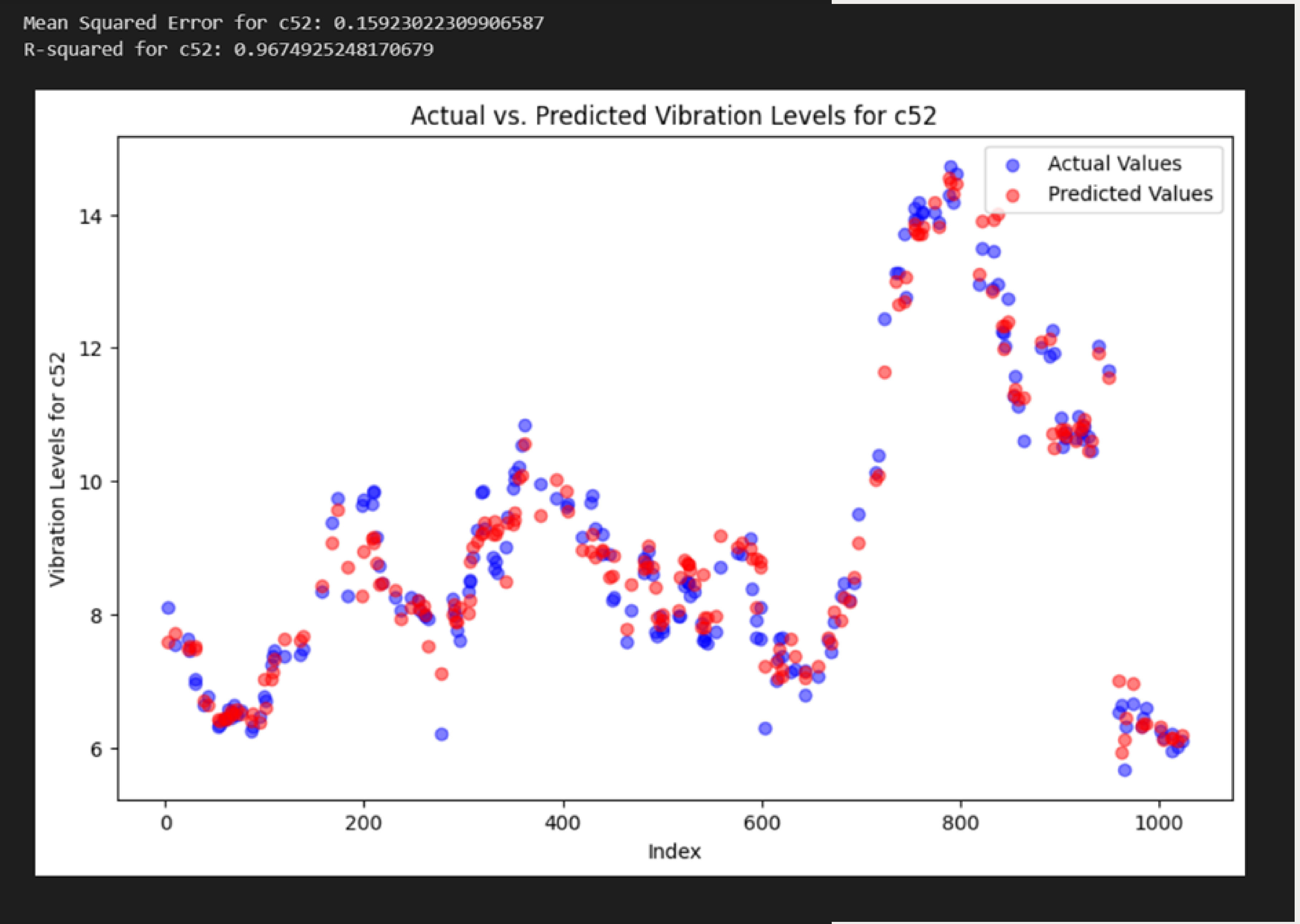
```
Mean Squared Error for c51: 0.4888078613912953
R-squared for c51: 0.9359045196858679
```



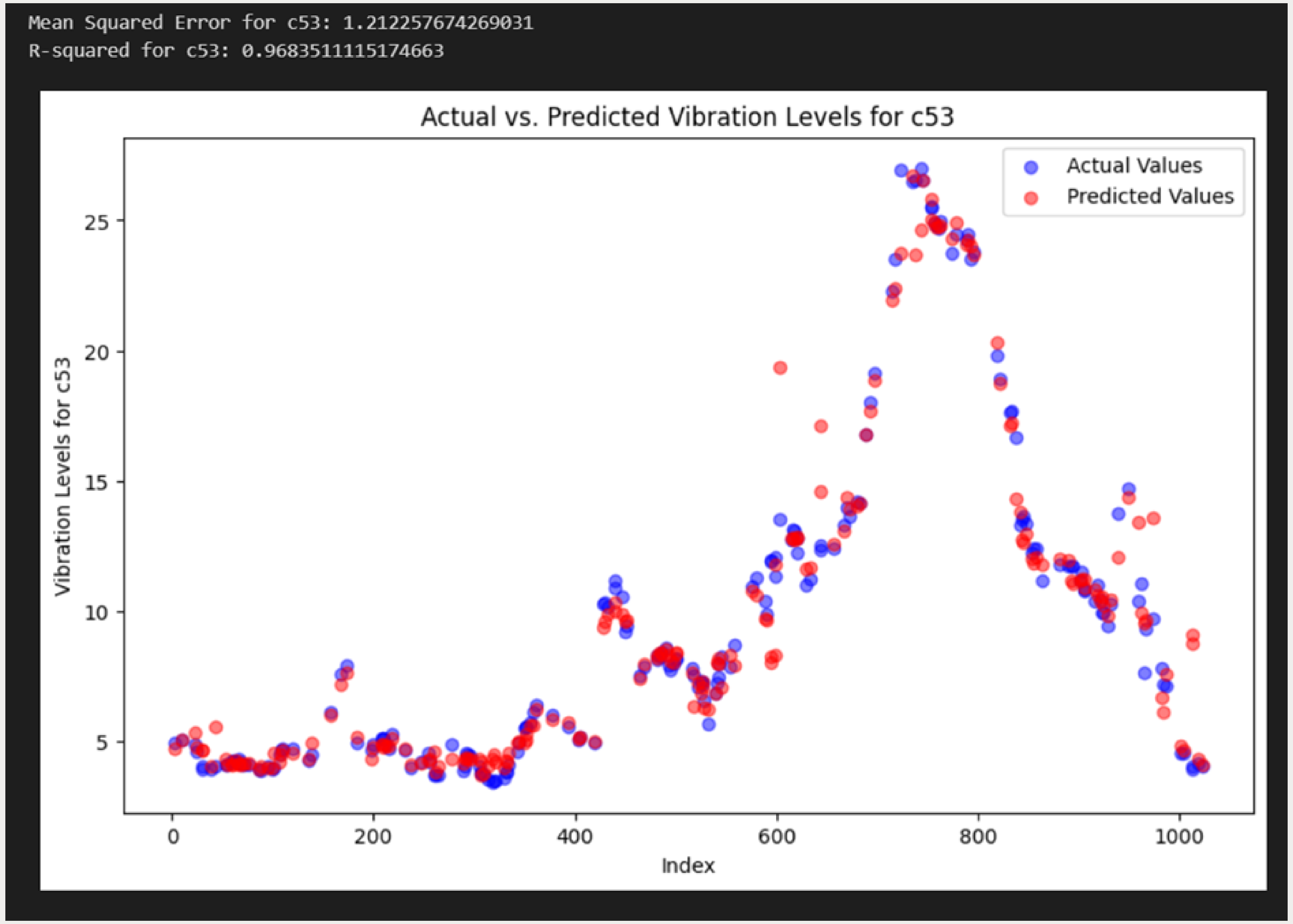


C51

On the left is the output for running MLR for c51. As we can see, this is not a good model.

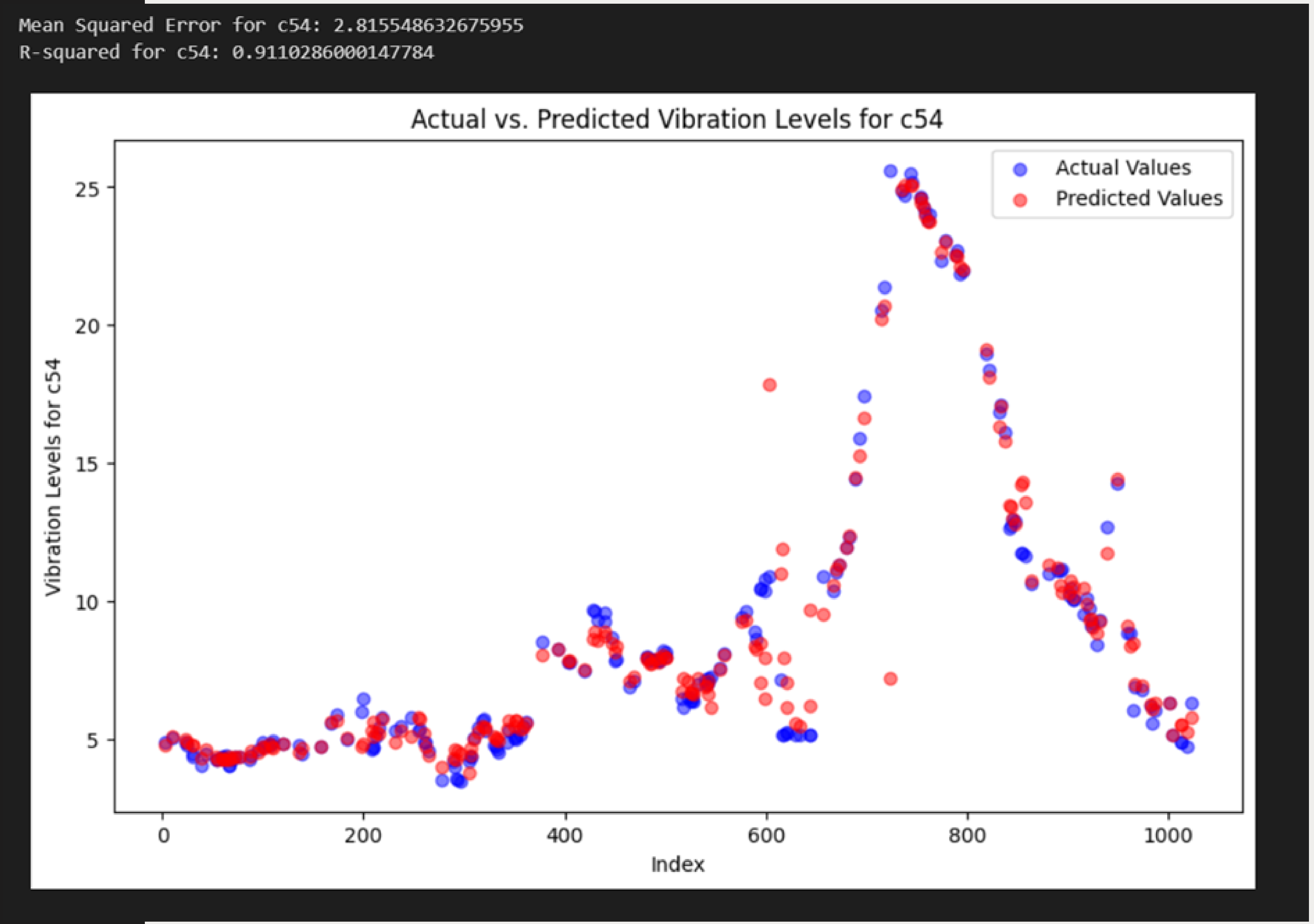


```
Mean Squared Error for c52: 0.15923022309906587
R-squared for c52: 0.9674925248170679
```



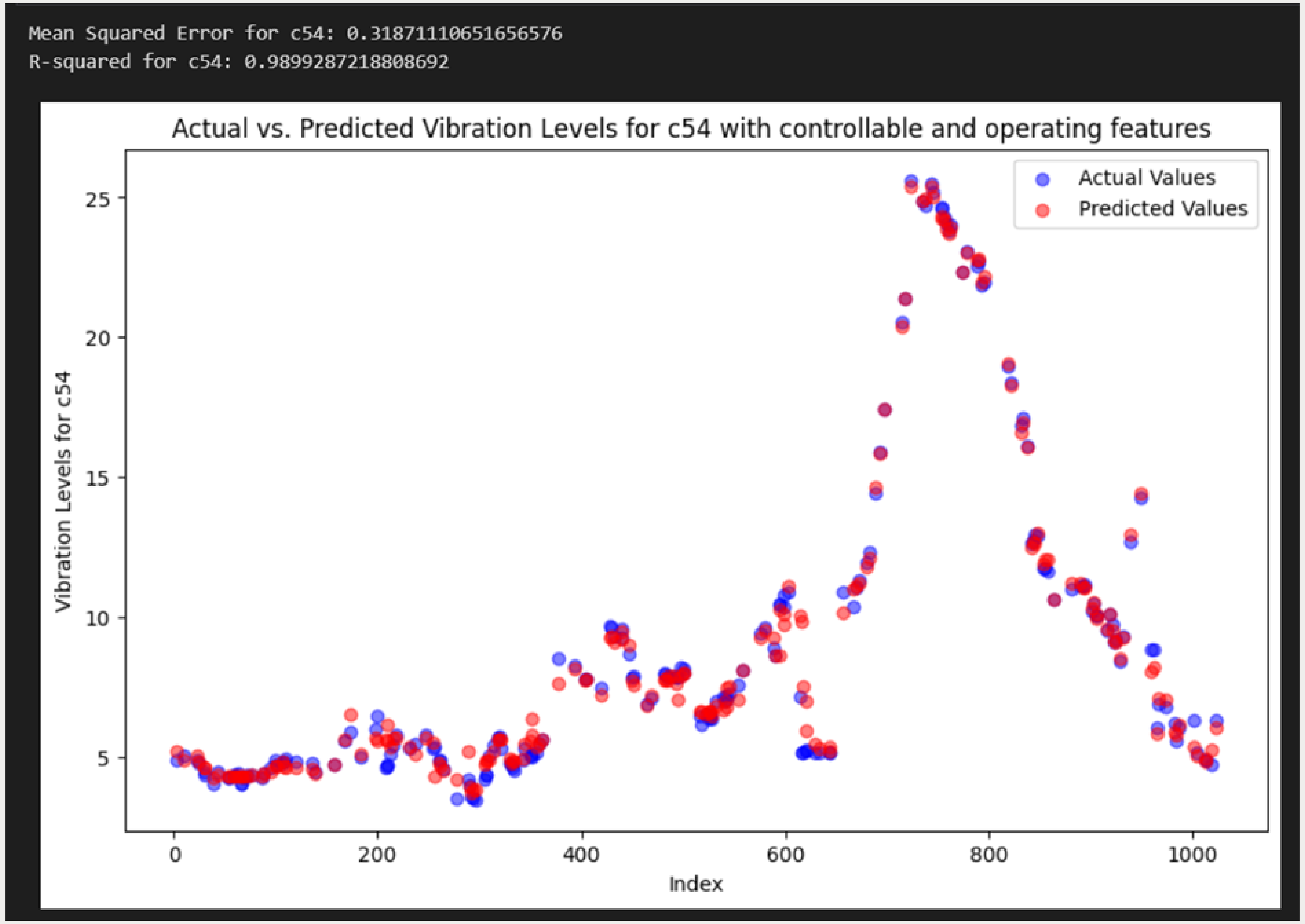
```
Mean Squared Error for c53: 1.212257674269031
R-squared for c53: 0.9683511115174663
```

based on controllable parameters...



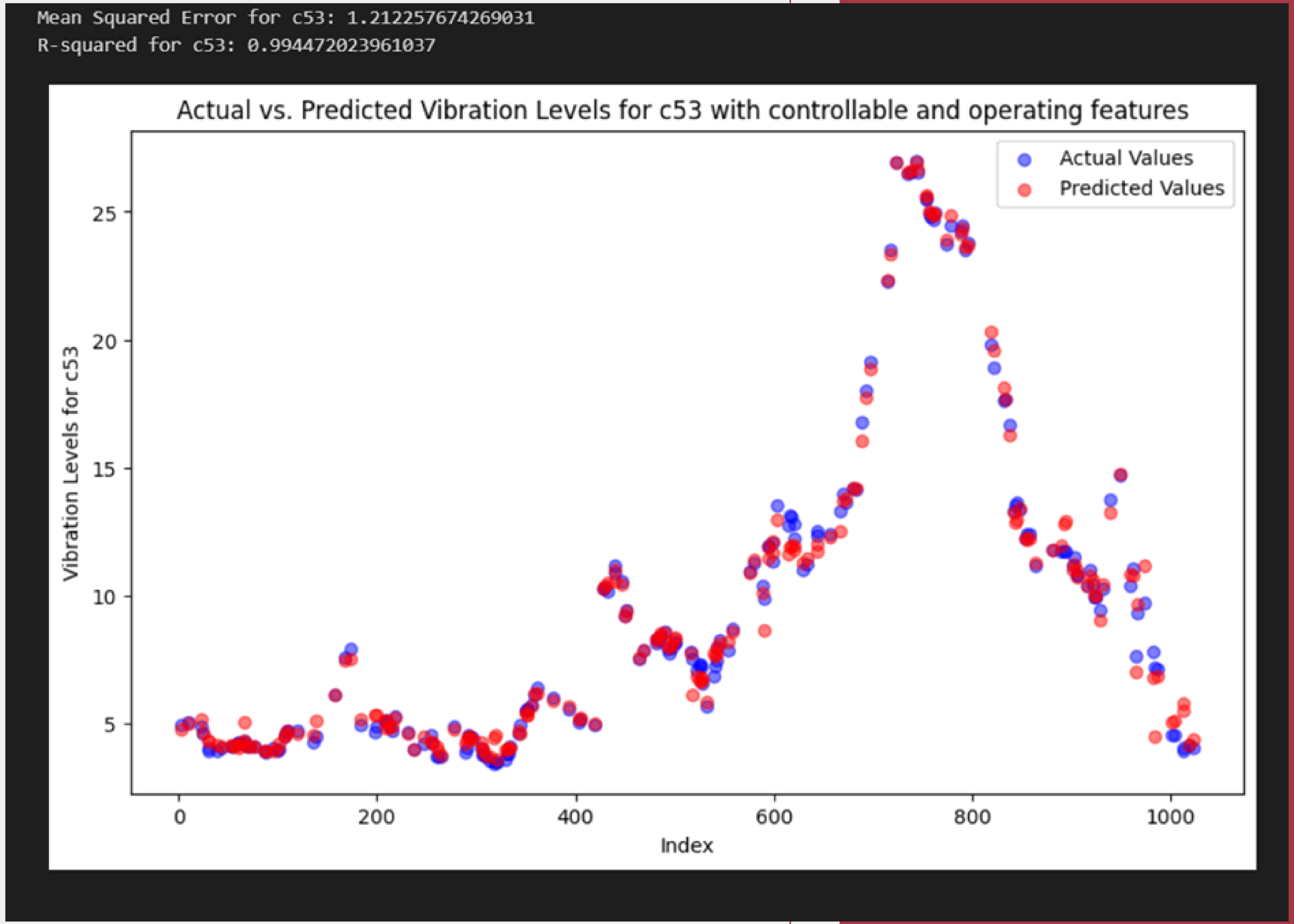
```
Mean Squared Error for c54: 2.815548632675955
R-squared for c54: 0.9110286000147784
```

Below are the figures for models c51, c52, c53, c54 when the features are **controllable and operating parameters**:



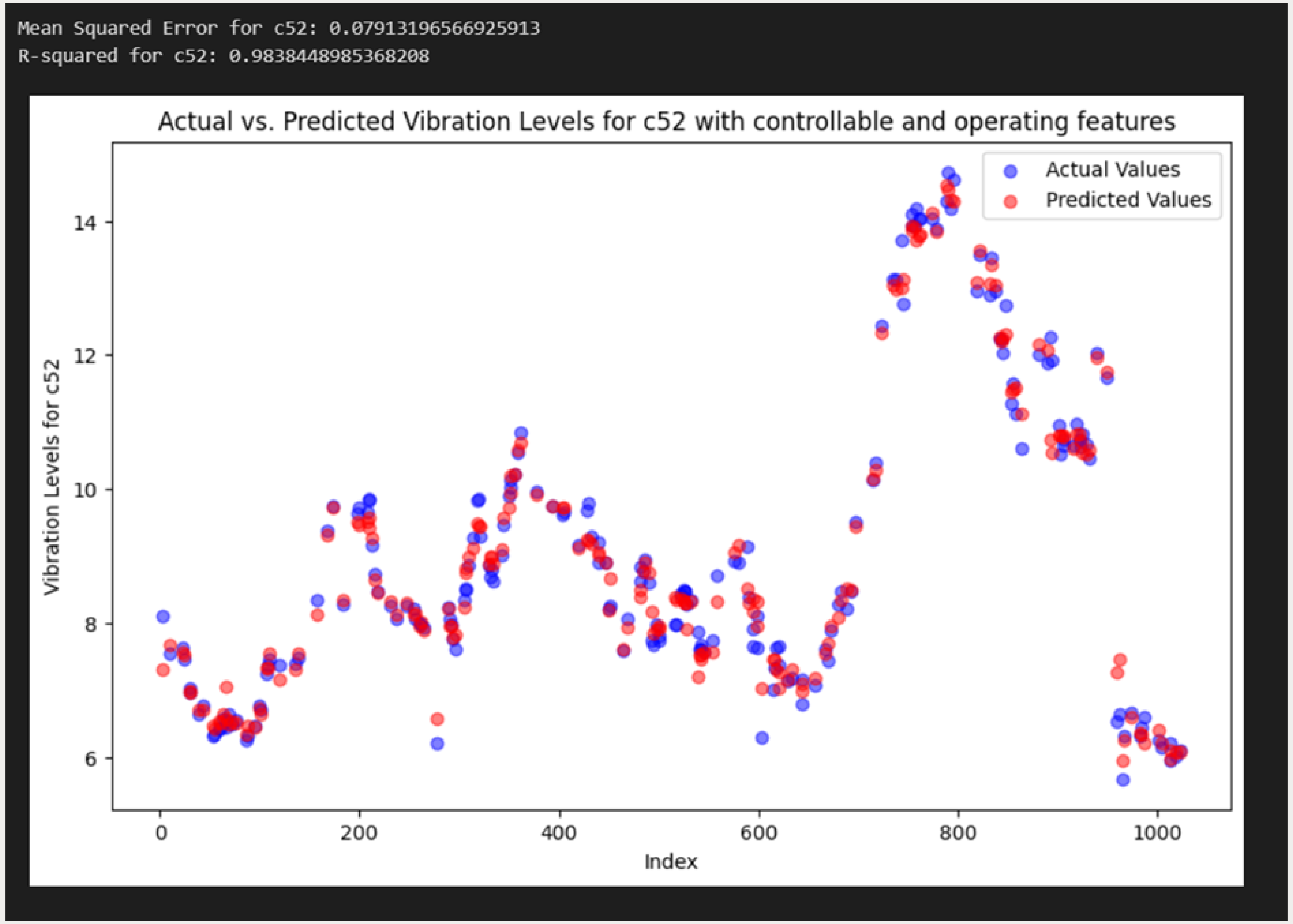
```
Mean Squared Error for c54: 0.31871110651656576
R-squared for c54: 0.9899287218808692
```

Actual vs. Predicted Vibration Levels for c54 with controllable and operating features
Actual Values
Predicted Values
Vibration Levels for c54
25
20
15
10
5
0
200
400
600
800
1000
Index

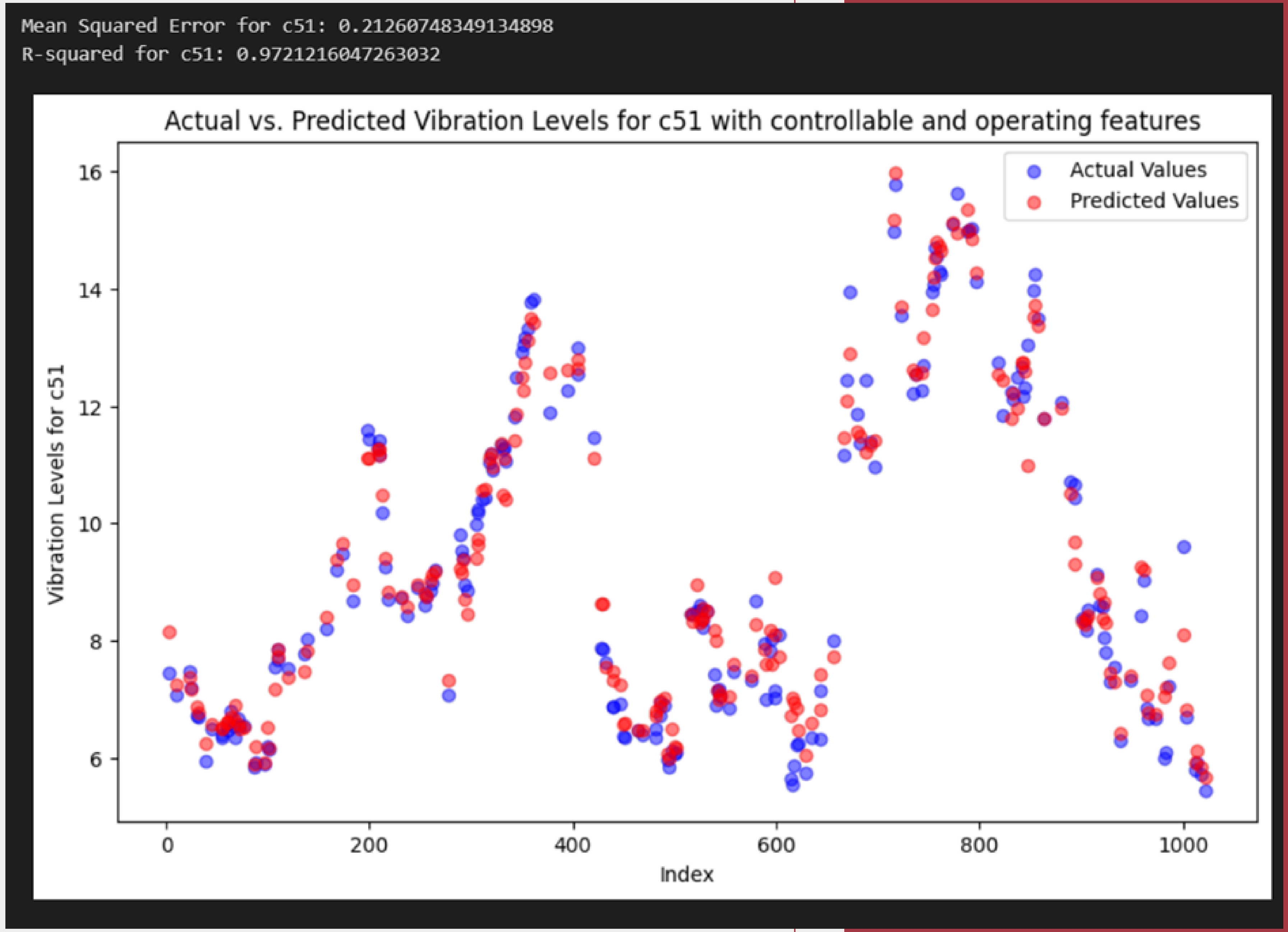


```
Mean Squared Error for c53: 1.212257674269031
R-squared for c53: 0.994472023961037
```

Actual vs. Predicted Vibration Levels for c53 with controllable and operating features
Actual Values
Predicted Values
Vibration Levels for c53
25
20
15
10
5
0
200
400
600
800
1000
Index



```
Mean Squared Error for c52: 0.07913196566925913
R-squared for c52: 0.9838448985368208
```



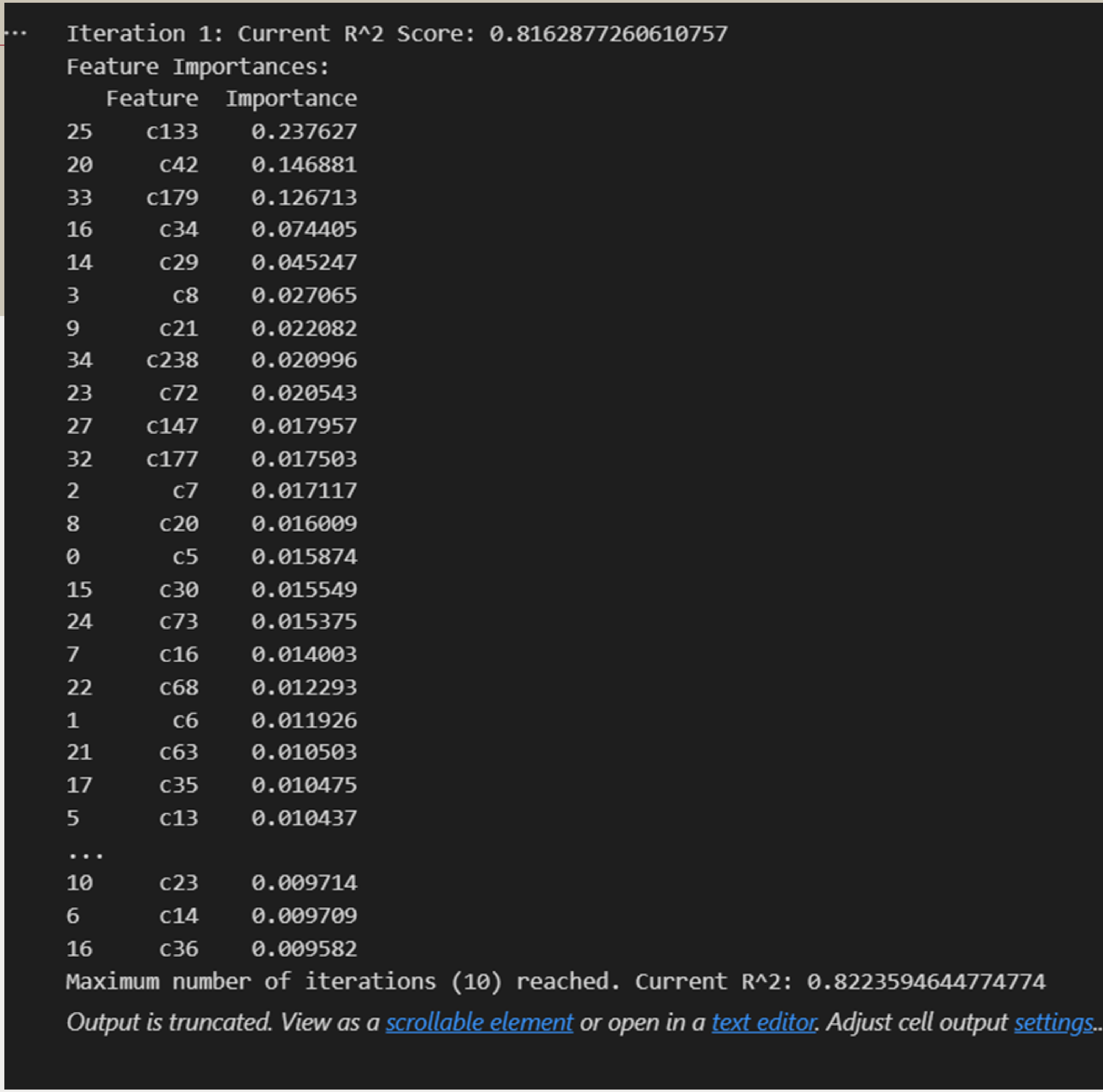
```
Mean Squared Error for c51: 0.21260748349134898
R-squared for c51: 0.9721216047263032
```

All these above 8 models were generated using random forrest regression. It uses tree based decision making and hence it has captured the trend of the parameter to model nicely.

C241

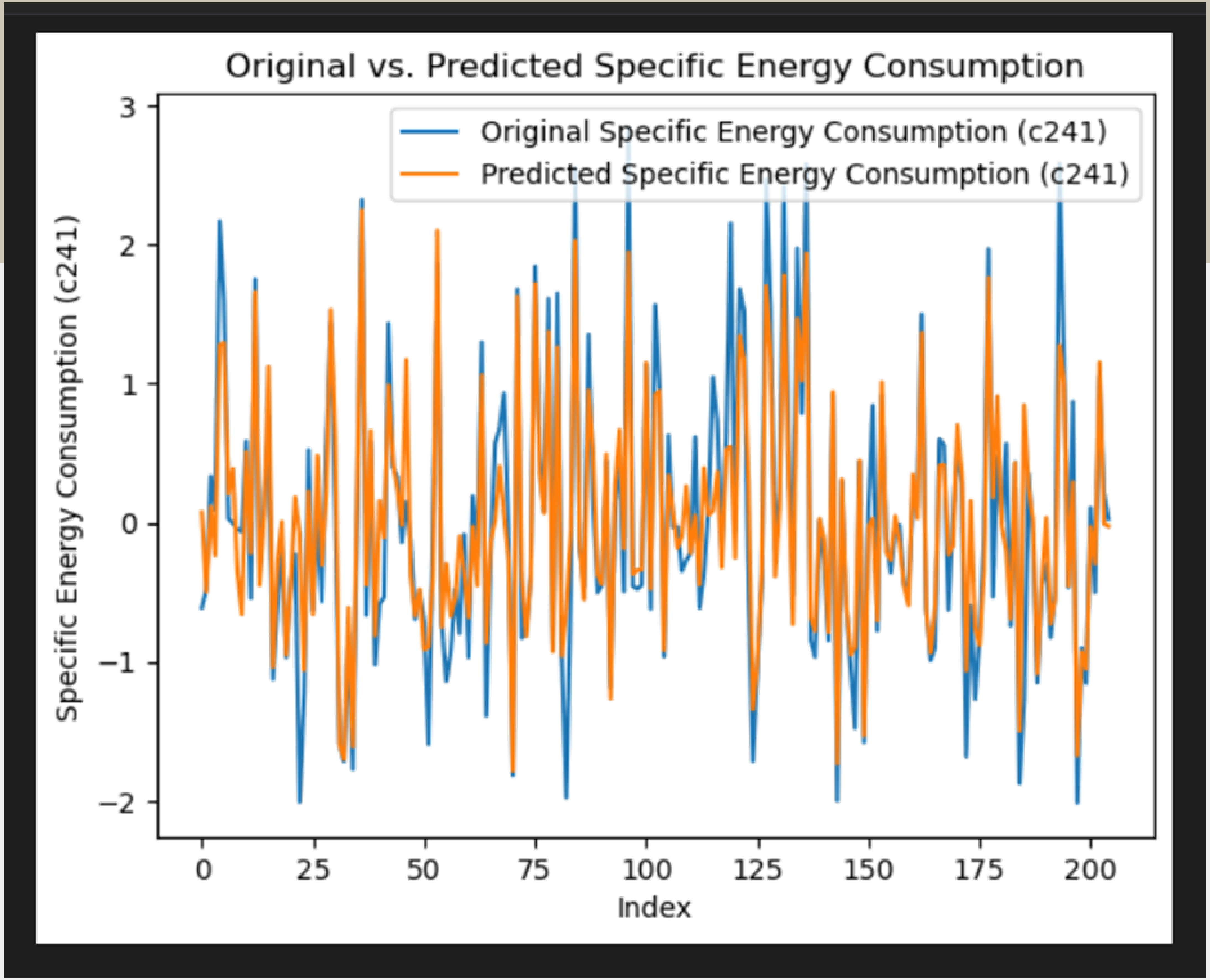
Again here we are using random forest regressor

We’ve used an 80-20 split and after each iteration if the threshold value of R square isn’t met, we drop the least important feature from the training and testing set, till the required R square value is met.



```
...  Iteration 1: Current R^2 Score: 0.8162877260610757
     Feature Importances:
         Feature  Importance
     25     c133    0.237627
     20      c42    0.146881
     33     c179    0.126713
     16      c34    0.074405
     14      c29    0.045247
     3        c8    0.027065
     9       c21    0.022082
     34     c238    0.020996
     23      c72    0.020543
     27     c147    0.017957
     32     c177    0.017503
     2        c7    0.017117
     8       c20    0.016009
     0        c5    0.015874
     15      c30    0.015549
     24      c73    0.015375
     7       c16    0.014003
     22      c68    0.012293
     1        c6    0.011926
     21      c63    0.010503
     17      c35    0.010475
     5       c13    0.010437
     ...
     10      c23    0.009714
     6       c14    0.009709
     16      c36    0.009582
     Maximum number of iterations (10) reached. Current R^2: 0.8223594644774774
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...


Original vs. Predicted Specific Energy Consumption
Original Specific Energy Consumption (c241)
Predicted Specific Energy Consumption (c241)
Specific Energy Consumption (c241)
3
2
1
0
−1
−2
0
25
50
75
100
125
150
175
200
Index

CHALLENGES

Cleaning the data- A lot of effort went into dealing with the non-numeric data and missing data, which was filled by the method of linear interpolation.

Removing columns with high correlation was tough. We used a factor called VIF to drop the columns which are highly correlated

Finding a model that works- The model created using MLR was not good as it did not fit well and gave us a very low R square value. So we used the Random Forest Regressor to create our models.

ACHIEVEMENTS

We dropped the highly correlated columns and trained the models for c51, c52, c53 and c54 based on the remaining columns.

We have trained the models on 80% of the data and tested it on the remaining 20%, and seen the R square and MSE values, as well as plotting the graph between observed values and predicted values. The R square value for our model is above the threshold of 0.8 and the MSE is relatively low. Graphically also the predicted and observed values have good agreement.

We correctly identified the levels of vibrations, i.e. Safe, moderate, high or critical based on the given thresholds.

We also trained an ML model to predict the specific energy using the random forest regressor.

RESULTS

The models trained for c51,c52,c53,c54 gave us a good R squared value (>0.9).

The R squared value for the specific energy model turned out to be 0.82 after dropping 10 columns.
hence, the minimum number of ‘independent’ variables that can be used to ‘only predict – not control’ the specific energy consumption is 25.