# DS - 203

# E11 - Data Analysis of a Chemical Processing Plant

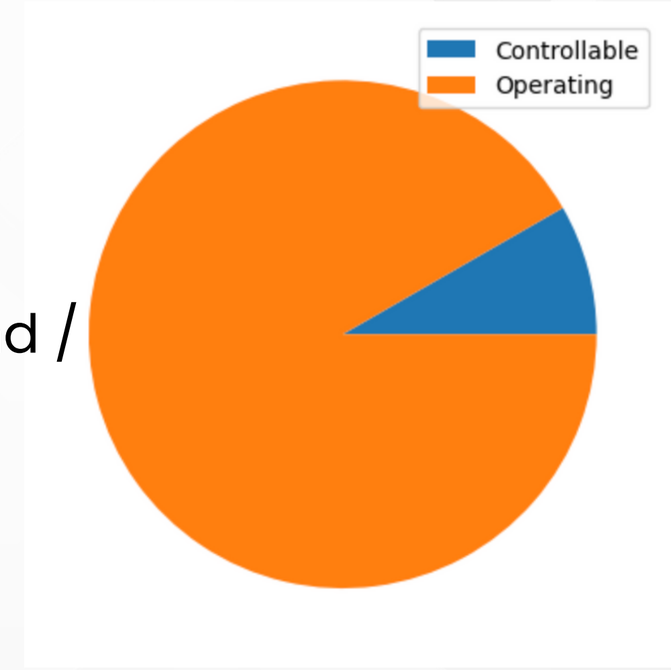
Saarthak Krishan 22B3959
Sanat Agrawal 22B3919
Yash Gupta 22B1813

# DATA DESCRIPTION

- The data of daily averaged values of observations logged by a data acquisition system at a chemical processing plant for 2 years and 9 months

- 241 various parameters that are monitored and / or controlled

- Two types of parameters : Operating and Controllable



# TASK

## Vibration of Equipments (c51, c52, c53, c54)

- To create ML models to predict the vibration levels
- Raise alerts and alarms if/when they reach HIGH and CRITICAL levels
- To create an automated vibration control and reduction system based on Controllable parameters

## Specific Energy (c241)

- To create ML models to predict the Specific Energy and find most significant parameters
- Find out the minimum number of ‘independent’ variables that can be used to ‘only predict – not control’

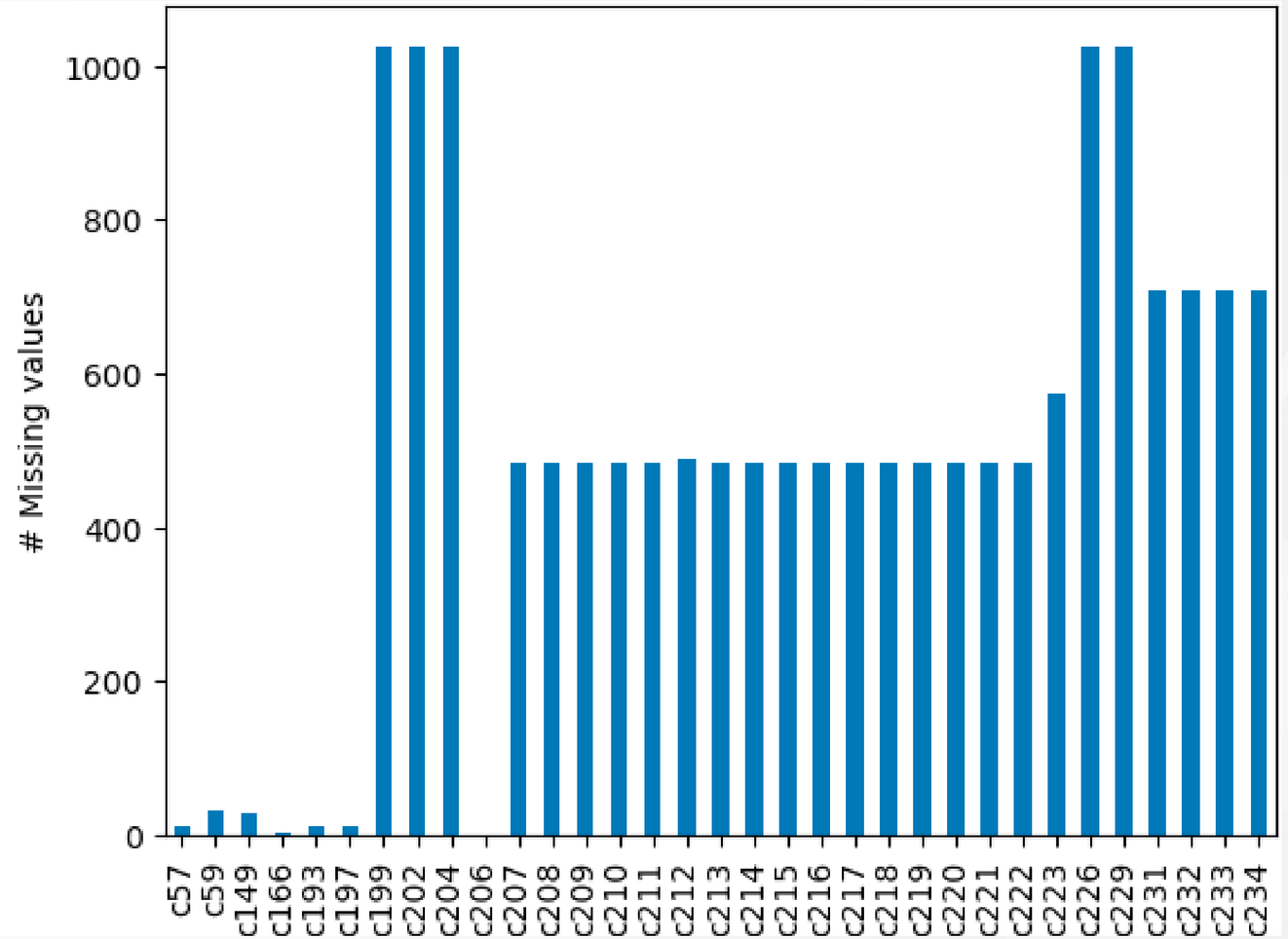
BUT, BEFORE THAT.....

# EDA & DATA CLEANING

- There are about 241 columns - But not all of them have valuable entries
- First priority before applying any ML model is to clean the data to choose our input variables wisely
- The major junk columns which can be removed include columns with-
  - Large number of missing data
  - Most of data being constant
- However, some columns with only a fraction of values missing can be managed by extrapolation of available data
- Also, 241 columns is a big number - fertile ground for multicollinearity!

# LARGE DATA MISSING

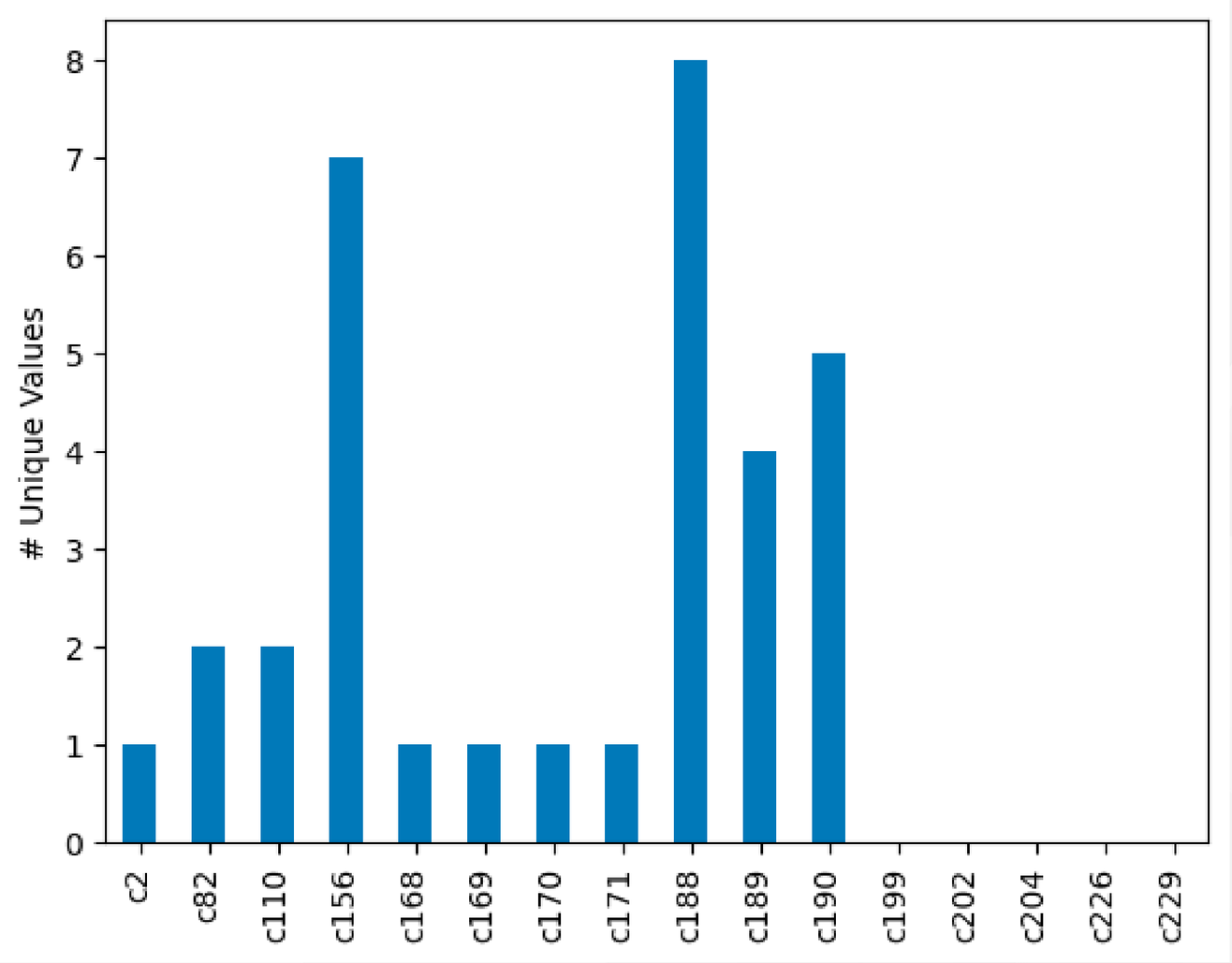
- The Graph alongside shows that some columns have high amount of data missing (more than half of the data)
- Thus, these columns don't really play an important role in predicting the parameters.
- Hence, They are best managed by dropping them.




**Columns dropped** - c199, c202, c204, c207, c208, c209, c210, c211, c212, c213, c214, c215, c216, c217, c218, c219, c220, 221, c222, c223, c226, c229, c231, c232, c233, c234

# CONSTANT DATA

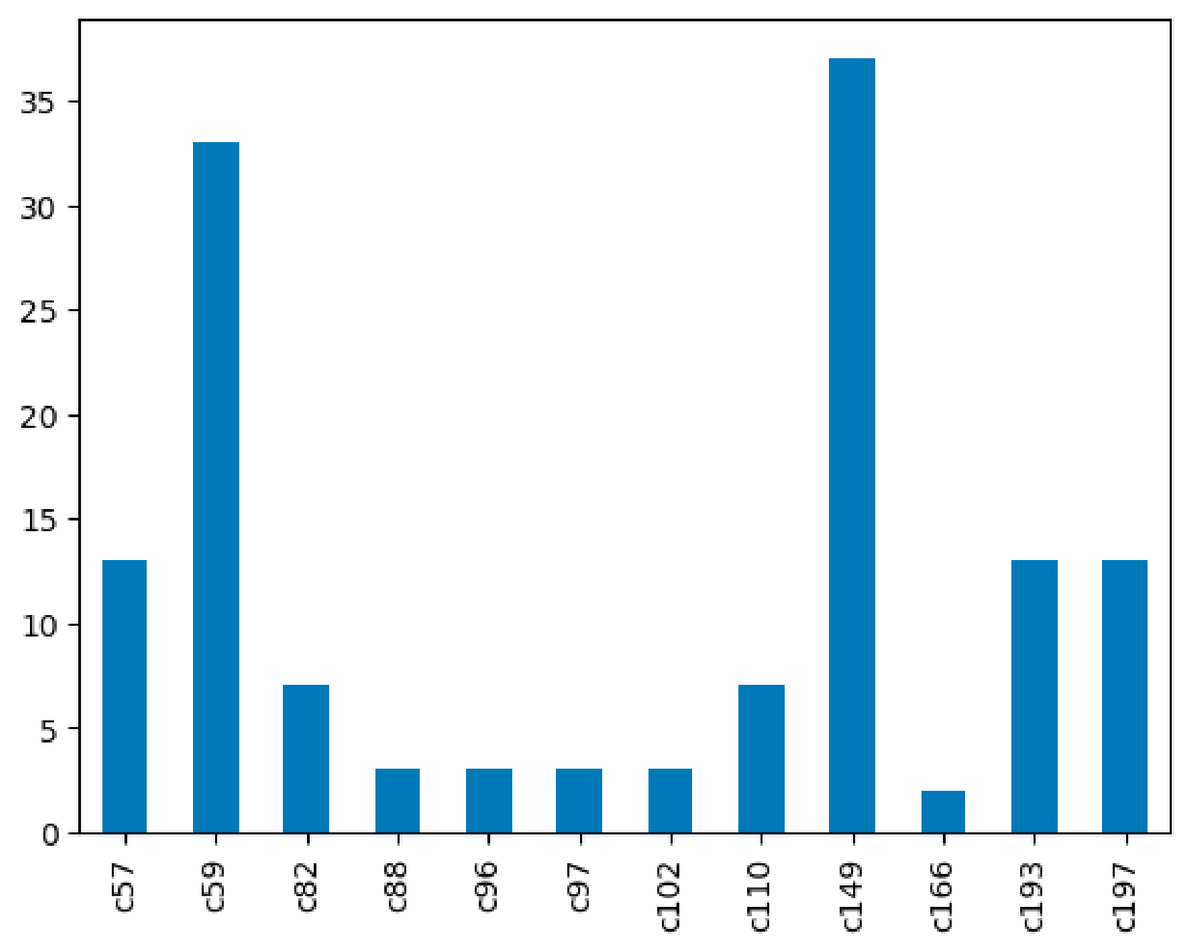
- The Graph alongside shows that some columns have very less unique values.
- This means most of the data in the column is constant
- This constant nature of the data is not very impactful for prediction and will be absorbed in the constant term of MLR. So we can drop them.




**Columns dropped** - c2, c82, c110, c156, c168, c169, c170, c171, c188, c189, c190

# SMALL DATA MISSING

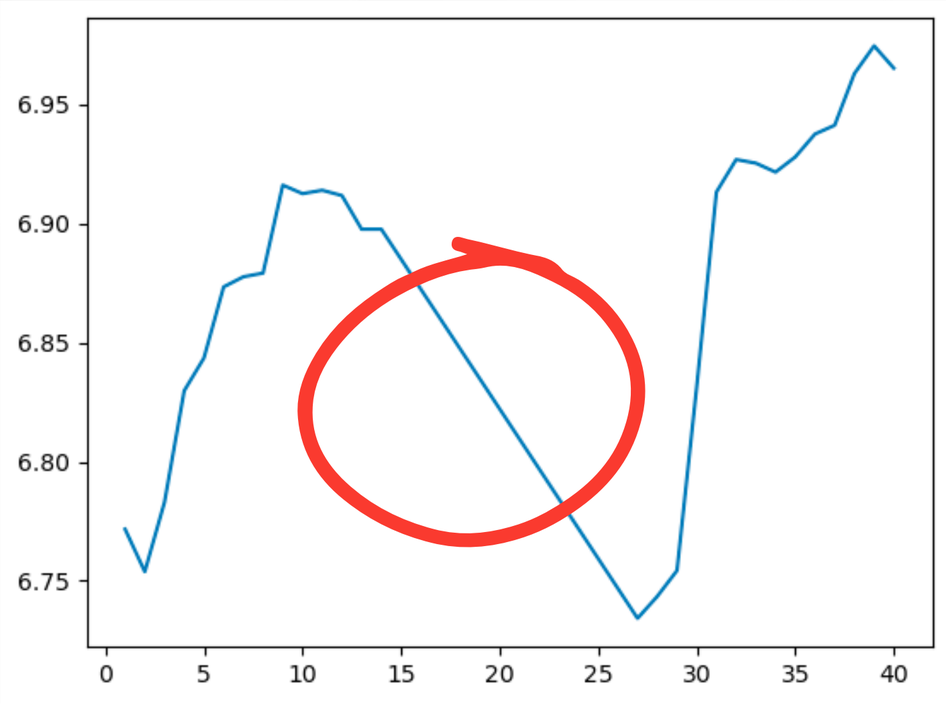
- The Graph alongside shows that some columns have very less values which are missing.
- It is hence, possible to first manage this missing data and then use it for the model.
- Managing was done by extrapolating the data based on the value of the data in its neighbourhood




**Columns managed** - c57, c59, c88, c96, c97, c102, c113, c133, c149, c166, c193, c197

# Ex: c57

```
manage=[]
first_nan = 563
numberofnan = 13
for i in range(numberofnan+2):
    manage.append(data['c57'][first_nan-1+i])
for i in range(numberofnan):
    data['c57'][first_nan+i]=np.round(manage[0]-(manage[0]-manage[numberofnan+1])/numberofnan*i,9)
```



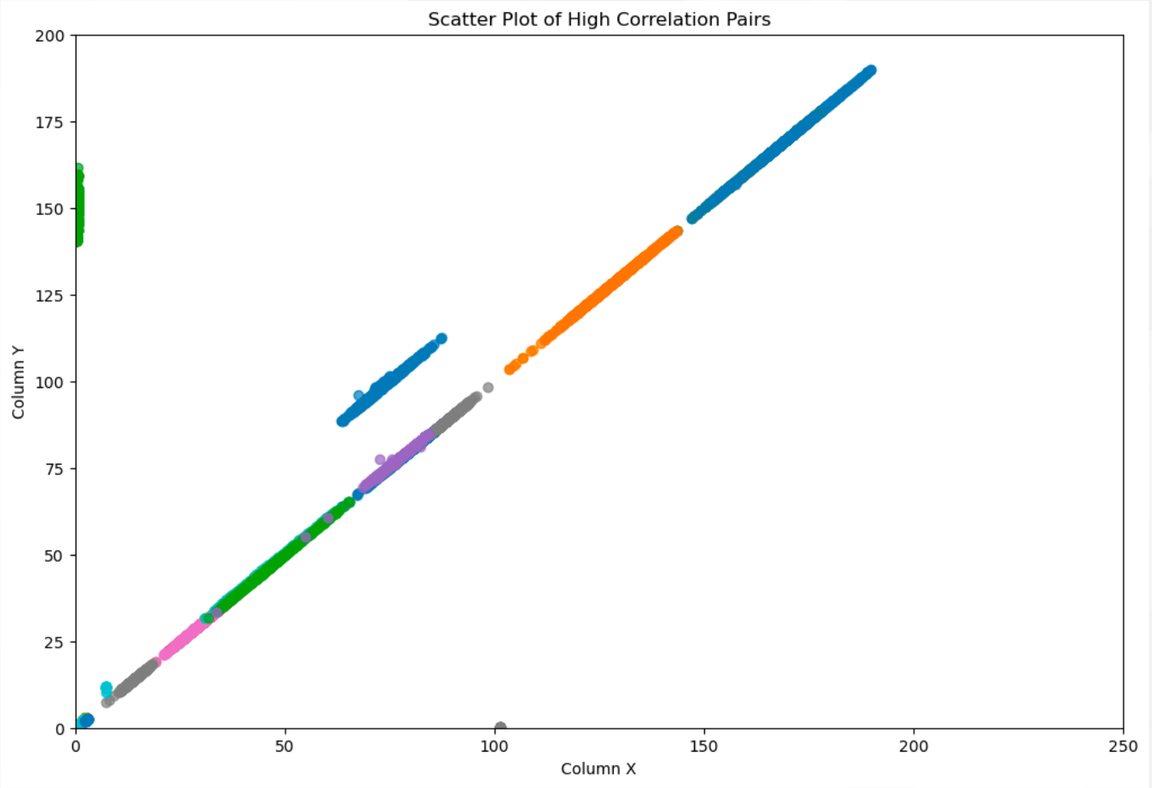

| | |
|---|---|
| 560 | 6.913851 |
| 561 | 6.911646 |
| 562 | 6.897517 |
| 563 | NaN |
| 564 | NaN |
| 565 | NaN |
| 566 | NaN |
| 567 | NaN |
| 568 | NaN |
| 569 | NaN |
| 570 | NaN |
| 571 | NaN |
| 572 | NaN |
| 573 | NaN |
| 574 | NaN |
| 575 | NaN |
| 576 | 6.734271 |
| 577 | 6.743500 |
| 578 | 6.754316 |

| | |
|---|---|
| 560 | 6.913851 |
| 561 | 6.911646 |
| 562 | 6.897517 |
| 563 | 6.897517 |
| 564 | 6.884960 |
| 565 | 6.872403 |
| 566 | 6.859845 |
| 567 | 6.847288 |
| 568 | 6.834730 |
| 569 | 6.822173 |
| 570 | 6.809615 |
| 571 | 6.797058 |
| 572 | 6.784500 |
| 573 | 6.771943 |
| 574 | 6.759386 |
| 575 | 6.746828 |
| 576 | 6.734271 |
| 577 | 6.743500 |
| 578 | 6.754316 |

# MULTICOLLINEARITY

- The columns are not independent, columns with high correlations will be dropped
- The central line in graph represents ~1 correlation coeff.

```
Columns c41 and c107 have correlation: 0.9999999999999997
Columns c46 and c184 have correlation: 0.9992806820646309
Columns c47 and c109 have correlation: 1.0000000000000002
Columns c48 and c111 have correlation: 0.9999999999999997
Columns c48 and c200 have correlation: 1.0000000000000002
Columns c48 and c227 have correlation: 1.0000000000000004
```




```python
correlation_matrix = X.corr()
corr_threshold = 0.99
high_corr_pairs = []
for i in range(len(correlation_matrix.columns)):
    for j in range(i+1, len(correlation_matrix.columns)):
        correlation = correlation_matrix.iloc[i, j]
        if (correlation) > corr_threshold:
            pair = (correlation_matrix.columns[i], correlation_matrix.columns[j], correlation)
            high_corr_pairs.append(pair)
for pair in high_corr_pairs:
    print(f"Columns {pair[0]} and {pair[1]} have correlation: {pair[2]}")
```

**Columns dropped** - c148, c69, c128, c24, c25, c136, c138, c139, c77, c144, c145, c104, c106, c105, c107, c184, c109, c111, c200, c227, c165, c112, c185, c114, c115, c119, c121, c83, c120, c122, c128, c141, c144, c120, c172, c200, c227, c228, c127, c130, c227

TASK 1

# PREDICTING THE VIBRATIONS

**The Vibration parameters are - c51, c52, c53, c54**

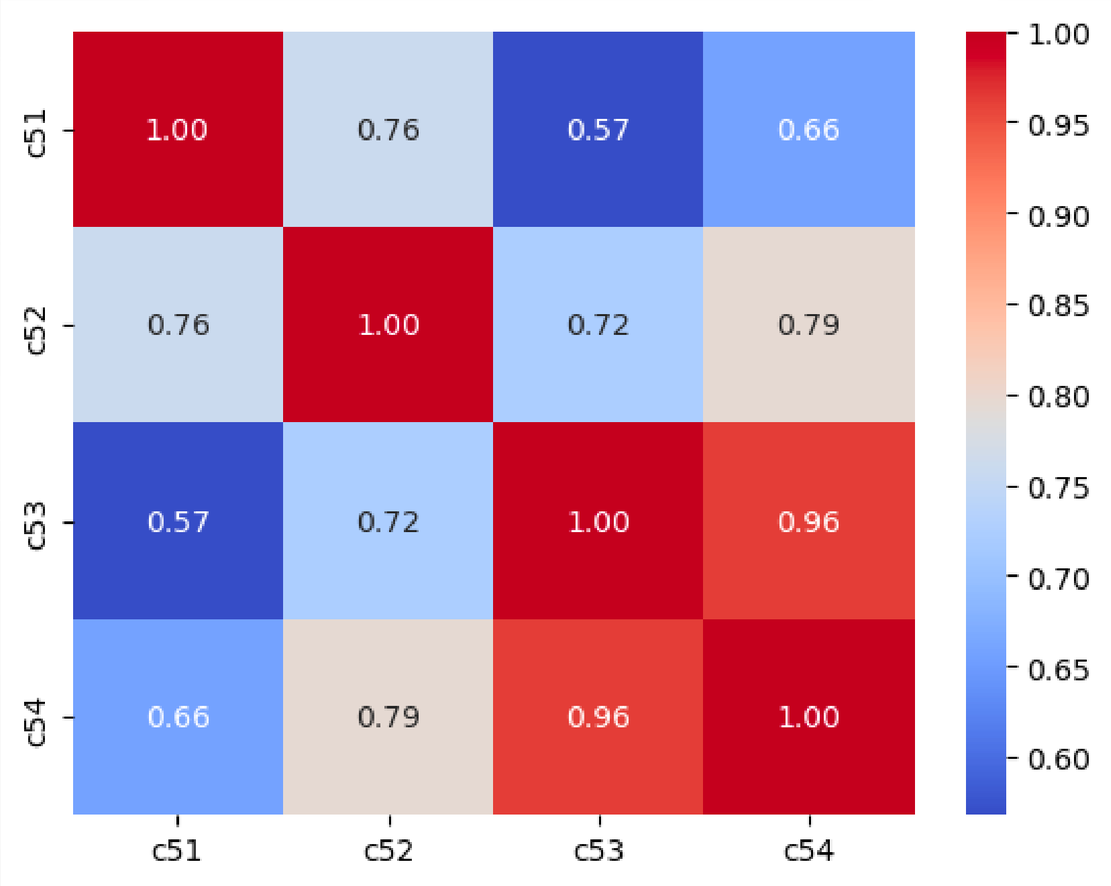
Q. Does this means we need to create 4 different ML models ??

A. Not really! Since these columns are not independent of each other. Hence some parameters can be predicted in terms of the other parameters.
This is justfied by the next slide.

TASK 1

# PREDICTING THE VIBRATIONS

## THE CORRELATION MATRIX




More the correlation, More the dependence.
Here we can see that the correlation of c53 and c54 is 0.96.
Hence predicting only one will give you the other.
So, it reduces our model count from 4 to 3.

# TASK 1
# PREDICTING THE VIBRATIONS
## SIMPLE LINEAR REGRESSION

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c54 | R-squared (uncentered): | 0.978 |
| Model: | OLS | Adj. R-squared (uncentered): | 0.978 |
| Method: | Least Squares | F-statistic: | 4.541e+04 |
| Date: | Sun, 12 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 21:14:26 | Log-Likelihood: | -1945.1 |
| No. Observations: | 1025 | AIC: | 3892. |
| Df Residuals: | 1024 | BIC: | 3897. |
| Df Model: | 1 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| c53 | 0.9165 | 0.004 | 213.088 | 0.000 | 0.908 | 0.925 |

| | | | |
|---|---|---|---|
| Omnibus: | 427.683 | Durbin-Watson: | 0.036 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 2367.630 |
| Skew: | -1.852 | Prob(JB): | 0.00 |
| Kurtosis: | 9.459 | Cond. No. | 1.00 |

We can apply SLR to fit c54 in terms of c53.
These are the results of the SLR. (Plots later)
We can see that :
value_c54=0.91*(value_c53)

TASK 1

# PREDICTING THE VIBRATIONS

## THE ML MODEL - PREDICTING C53

- Now that we are done with the prerequisites, we can finally move on to creating an ML model to predict our parameters.
- The main steps include:
  - Splitting the data into train (80%) and test (20%)
  - Predicting using the training data by **Multiple Linear Regression**
  - Dropping variables with high p-value (more than 0.05) to get better results
  - Checking ther prediction with the testing data

TASK 1

# PREDICTING THE VIBRATIONS

## KEY FEATURE : DROPPING MECHANISM

```
dropped_vars = []
R2_values = []
dropped_vars = []
R2_values = []
X=sm.add_constant(X)
X_train = X_train.apply(pd.to_numeric, errors='coerce')
while len(X_train.columns) > 1:
    model = sm.OLS(y_train, X_train).fit()
    max_p_value = model.pvalues[1:].max()

    if max_p_value > 0.05:
        dropped_var = model.pvalues[1:].idxmax()
        X_train = X_train.drop(columns=[dropped_var])
        model = sm.OLS(y_train, X_train).fit()
        dropped_vars.append(dropped_var)
        R2_values.append(model.rsquared)
    else:
        break

summary_table = pd.DataFrame({
    "Dropped Variable": dropped_vars,
    "R2": R2_values
})
```

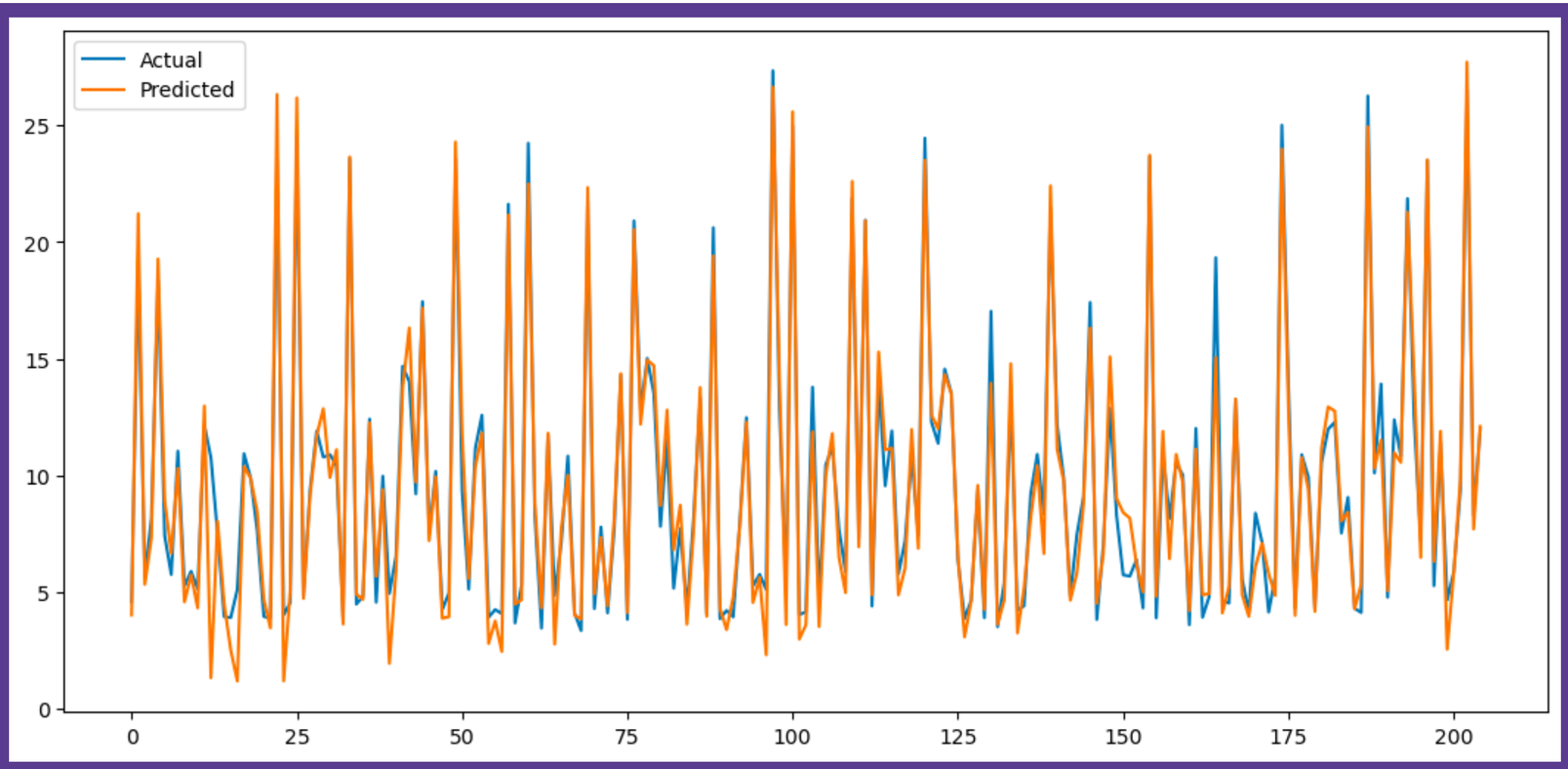
- Creates a model and find the p-values
- Removes column with biggest p-value if it’s more than 0.05
- Repeats till all p-values are <0.05
- Stores the dropped variables for future reference

# ML MODEL

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c53 | R-squared: | 0.984 |
| Model: | OLS | Adj. R-squared: | 0.982 |
| Method: | Least Squares | F-statistic: | 447.3 |
| Date: | Mon, 13 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 02:46:57 | Log-Likelihood: | -1009.7 |
| No. Observations: | 820 | AIC: | 2217. |
| Df Residuals: | 721 | BIC: | 2684. |
| Df Model: | 98 | | |
| Covariance Type: | nonrobust | | |

We have our ML model ready for further use.
It can be used by the client to predict the value of the vibration parameter by inputting other parameters.
Now, lets proceed to the next step.....

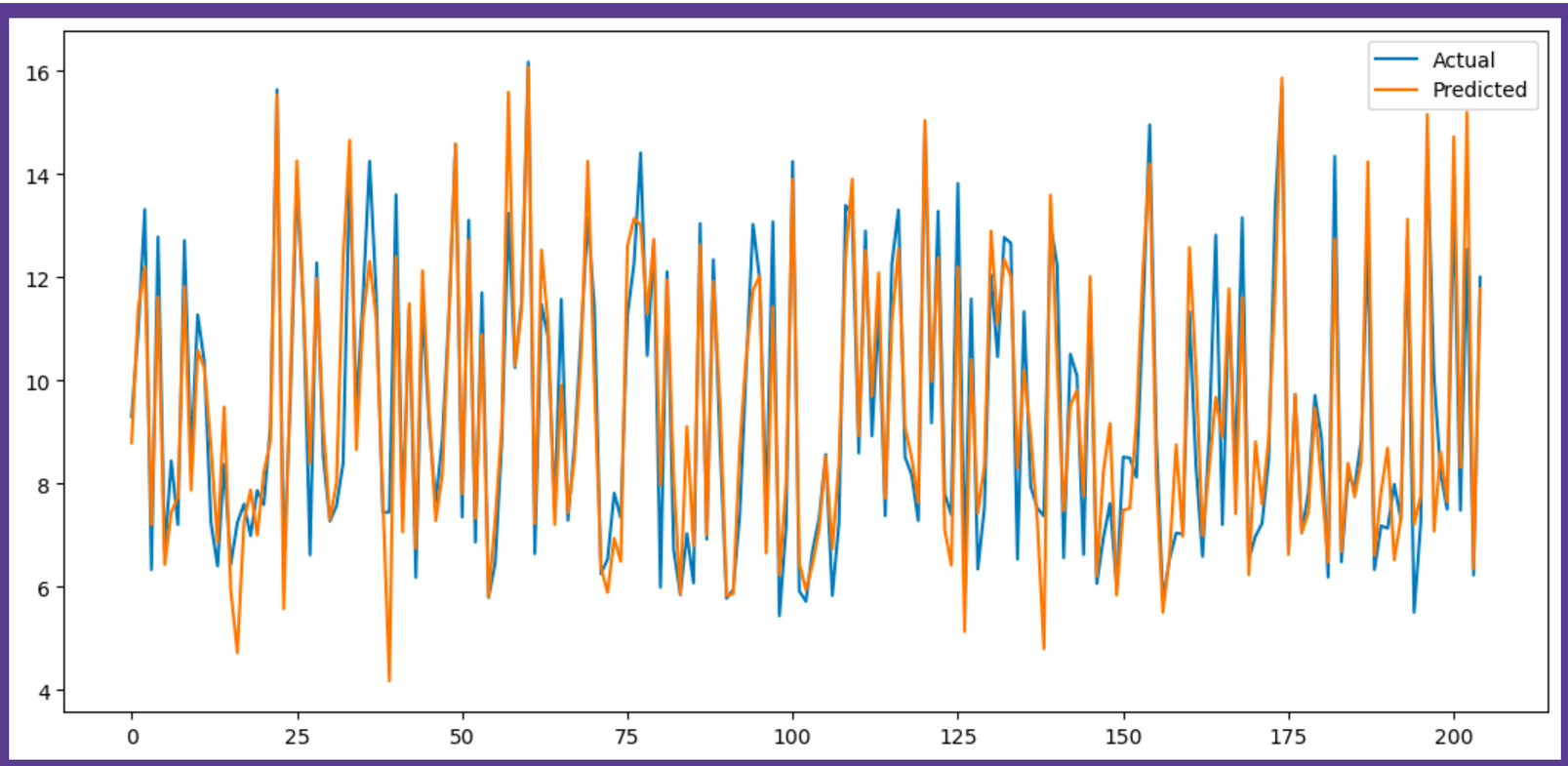


# ML MODEL

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c51 | R-squared: | 0.933 |
| Model: | OLS | Adj. R-squared: | 0.924 |
| Method: | Least Squares | F-statistic: | 108.7 |
| Date: | Mon, 13 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 05:22:59 | Log-Likelihood: | -875.96 |
| No. Observations: | 820 | AIC: | 1940. |
| Df Residuals: | 726 | BIC: | 2383. |
| Df Model: | 93 | | |
| Covariance Type: | nonrobust | | |

Similarly, for c51....

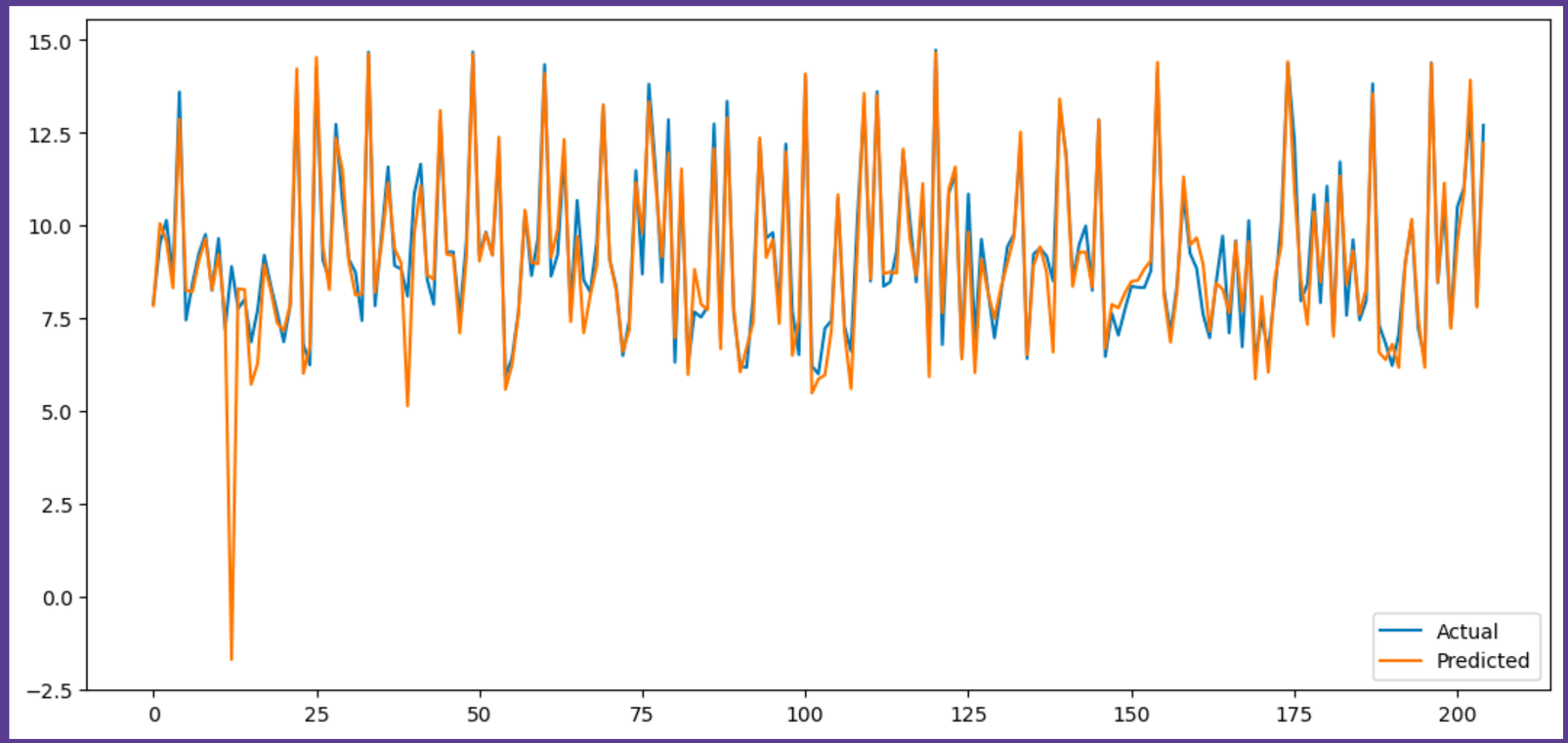


# ML MODEL

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c52 | R-squared: | 0.961 |
| Model: | OLS | Adj. R-squared: | 0.957 |
| Method: | Least Squares | F-statistic: | 194.9 |
| Date: | Mon, 13 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 05:28:12 | Log-Likelihood: | -481.64 |
| No. Observations: | 820 | AIC: | 1151. |
| Df Residuals: | 726 | BIC: | 1594. |
| Df Model: | 93 | | |
| Covariance Type: | nonrobust | | |

Similarly, for c52....

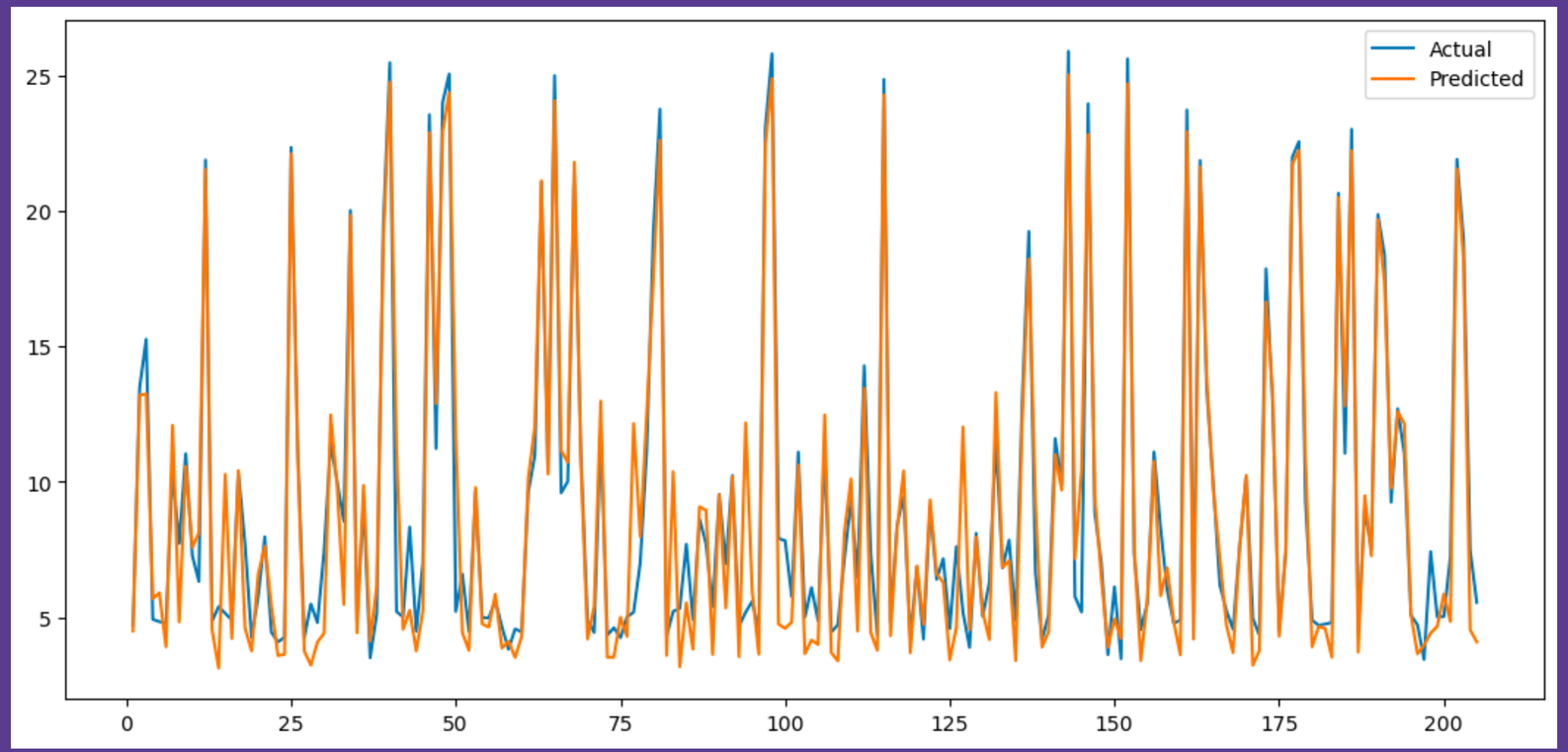


# ML MODEL

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c54 | R-squared (uncentered): | 0.978 |
| Model: | OLS | Adj. R-squared (uncentered): | 0.978 |
| Method: | Least Squares | F-statistic: | 3.684e+04 |
| Date: | Mon, 13 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 13:22:36 | Log-Likelihood: | -1546.6 |
| No. Observations: | 820 | AIC: | 3095. |
| Df Residuals: | 819 | BIC: | 3100. |
| Df Model: | 1 | | |
| Covariance Type: | nonrobust | | |

And predicted c54 from c53....



TASK 2

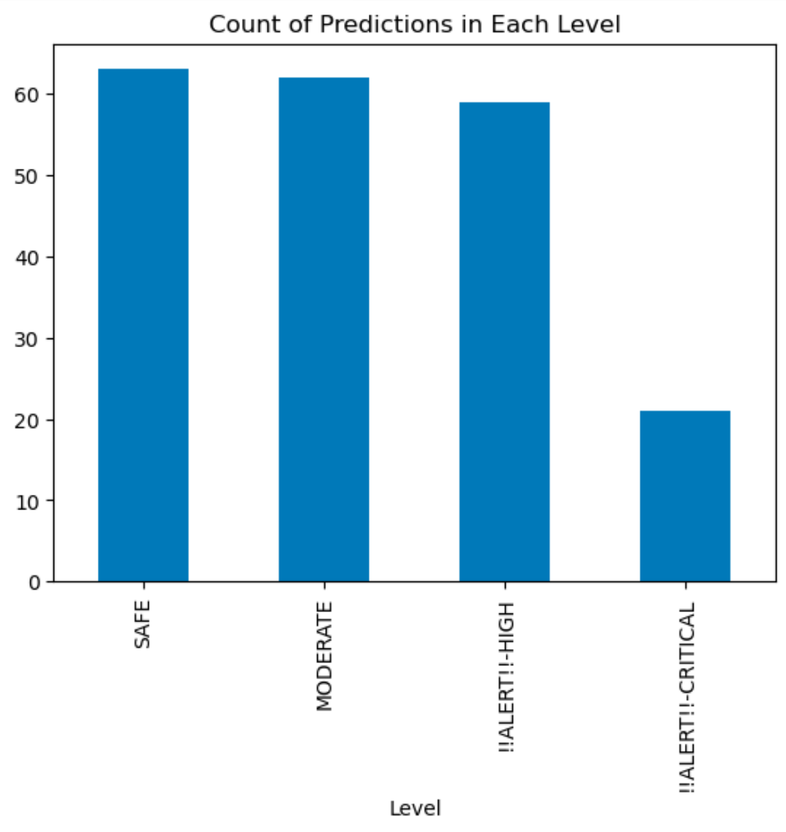
# CLASSIFICATION OF DATA

- The next task is to classify the parameters into the following:
  - Safe
  - Moderate
  - High
  - Critical
- We need to alarm the client when the output is in the high and critical region
- In the upcoming slides we will also make an automated process which will tune down the input parameters to bring the output to the safe range.

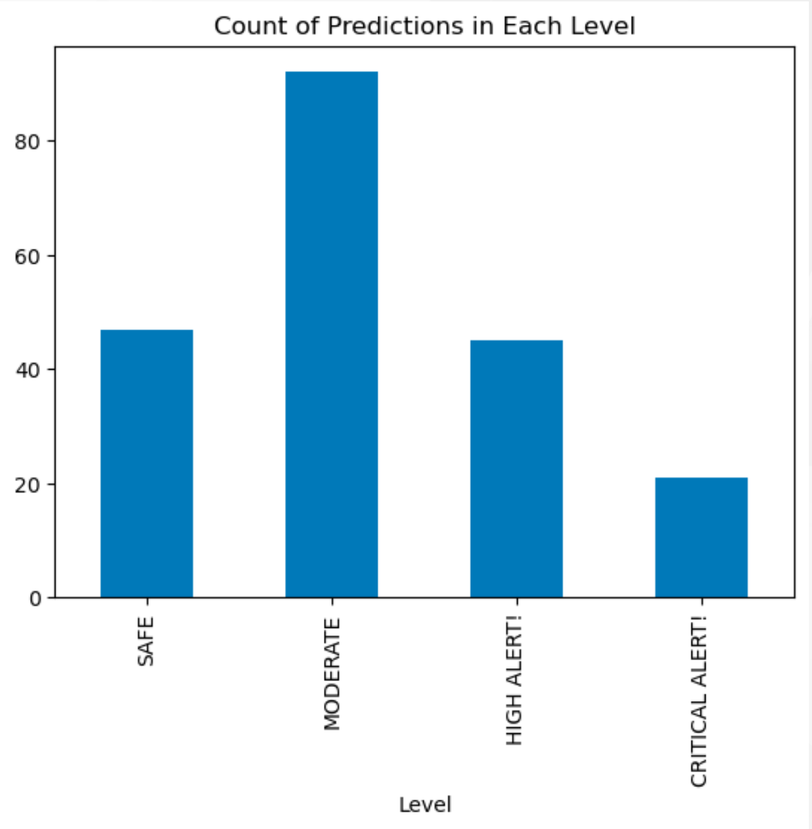
TASK 2

# CLASSIFICATION OF DATA

| Predicted | Level |
|---|---|
| 4.029122 | SAFE |
| 21.219929 | !!ALERT!!-CRITICAL |
| 5.339619 | MODERATE |
| 7.327754 | MODERATE |
| 19.279885 | !!ALERT!!-HIGH |
| 8.943032 | MODERATE |
| 6.678795 | MODERATE |
| 10.316201 | !!ALERT!!-HIGH |
| 4.590076 | SAFE |
| 5.744811 | MODERATE |
| 4.330032 | SAFE |
| 12.983177 | !!ALERT!!-HIGH |
| 1.338044 | SAFE |
| 8.053016 | MODERATE |
| 4.511980 | SAFE |




Classification on training Data




Classification on testing Data

## TASK 3

# DEPENDENCE ON CONTROLLABLE PARAMETERS

- Now, we repeat Task 1 but this time with only the controllable parameters.
- Note that this is not to get a perfect predicting ML model but to get an idea of the sigificance of each controllable parameter on the output.
- This knowledge, in turn, will be used to tune our parameters to change the output as desired.

TASK 3

# DEPENDENCE ON CONTROLLABLE PARAMETERS

## THE ML MODEL - PREDICTING C53

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c53 | R-squared (uncentered): | 0.948 |
| Model: | OLS | Adj. R-squared (uncentered): | 0.947 |
| Method: | Least Squares | F-statistic: | 915.7 |
| Date: | Mon, 13 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 03:45:57 | Log-Likelihood: | -1973.4 |
| No. Observations: | 820 | AIC: | 3979. |
| Df Residuals: | 804 | BIC: | 4054. |
| Df Model: | 16 | | |
| Covariance Type: | nonrobust | | |

Notice how how the metrics like Rsq and F-statistics are not that good. This was expected since we are not interested in predicting but finding impact of parameters on the output.

TASK 3

# DEPENDENCE ON CONTROLLABLE PARAMETERS

| | |
|---|---|
| c30 | 3.670999 |
| c155 | 0.663373 |
| c31 | 0.616908 |
| c33 | 0.577768 |
| c143 | 0.497452 |
| c26 | 0.365595 |
| c28 | 0.315583 |
| c163 | 0.050352 |
| c160 | −0.002877 |
| c157 | −0.127678 |
| c29 | −0.284599 |
| c139 | −0.363329 |
| c156 | −0.971483 |
| c27 | −1.032121 |
| c142 | −1.370942 |
| c39 | −8.805813 |

This table shows the value of coefficient of the controllable parameters in descending order. More the coefficient, more impact will they cause on the output when changed.

Hence, for c53, its best to tune c30 to change the output.
A similar analysis can be done for other vibrations parameters as well.

TASK 4

# MANAGING HIGH & CRITICAL VALUES

We have made an automated process which automatically tunes down the value of the column c30, so as to decrease the value of the output whenever an Alert is raised.

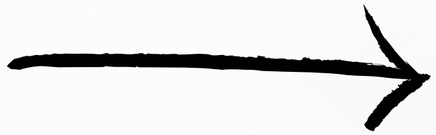
```
epsilon=0.01
threshold=10 # Change as desired
for i in high_alert_indices:
    while (model.predict(X_test.iloc[i]) > threshold).any():
        X_test.loc[i, 'c30'] -= epsilon
for i in critical_alert_indices:
    while (model.predict(X_test.iloc[i]) > threshold).any():
        X_test.loc[i, 'c30'] -= epsilon
```

TASK 4

# MANAGING HIGH & CRITICAL VALUES

## RESULTS

| | |
|---|---|
| 14.342545 | HIGH ALERT! |
| 3.185929 | SAFE |
| 13.140587 | HIGH ALERT! |
| 7.466948 | MODERATE |
| 6.920808 | MODERATE |
| 4.988905 | SAFE |
| 19.671613 | HIGH ALERT! |
| 3.792722 | SAFE |
| 14.777273 | HIGH ALERT! |
| 5.780866 | MODERATE |
| 8.244789 | MODERATE |
| 9.703066 | MODERATE |
| 2.555634 | SAFE |
| 4.686578 | SAFE |
| 6.174865 | MODERATE |
| 8.839748 | MODERATE |
| 4.638906 | SAFE |
| 3.062260 | SAFE |
| 21.905139 | CRITICAL ALERT! |
| 15.076021 | HIGH ALERT! |
| 2.644733 | SAFE |

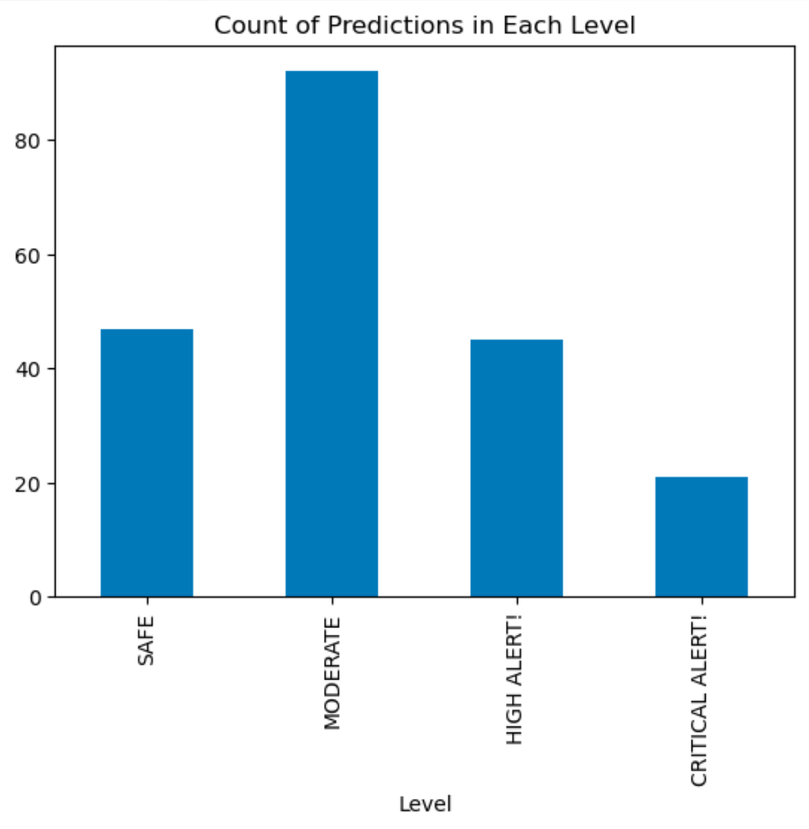


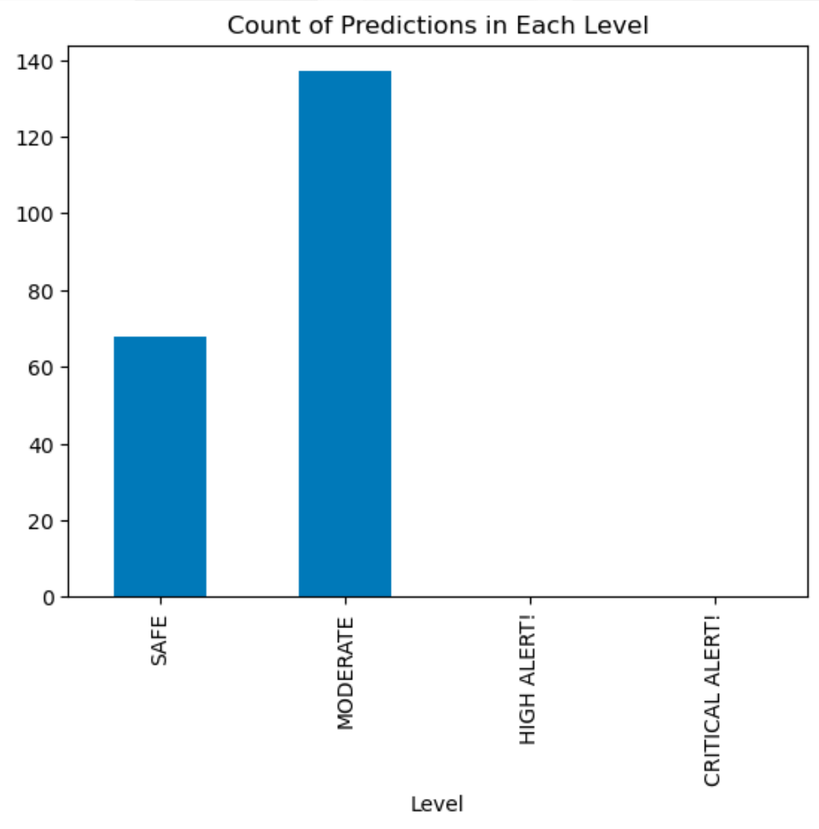
| | |
|---|---|
| 9.974056 | MODERATE |
| 3.185929 | SAFE |
| 9.983527 | MODERATE |
| 7.466948 | MODERATE |
| 6.920808 | MODERATE |
| 4.988905 | SAFE |
| 9.980175 | MODERATE |
| 3.792722 | SAFE |
| 9.968264 | MODERATE |
| 5.780866 | MODERATE |
| 8.244789 | MODERATE |
| 9.703066 | MODERATE |
| 2.555634 | SAFE |
| 4.686578 | SAFE |
| 6.174865 | MODERATE |
| 8.839748 | MODERATE |
| 4.638906 | SAFE |
| 3.062260 | SAFE |
| -31.668026 | SAFE |
| 9.973332 | MODERATE |
| 2.644733 | SAFE |

TASK 4

# MANAGING HIGH & CRITICAL VALUES

## RESULTS



TASK 5.1

# PREDICTING SPECIFIC ENERGY

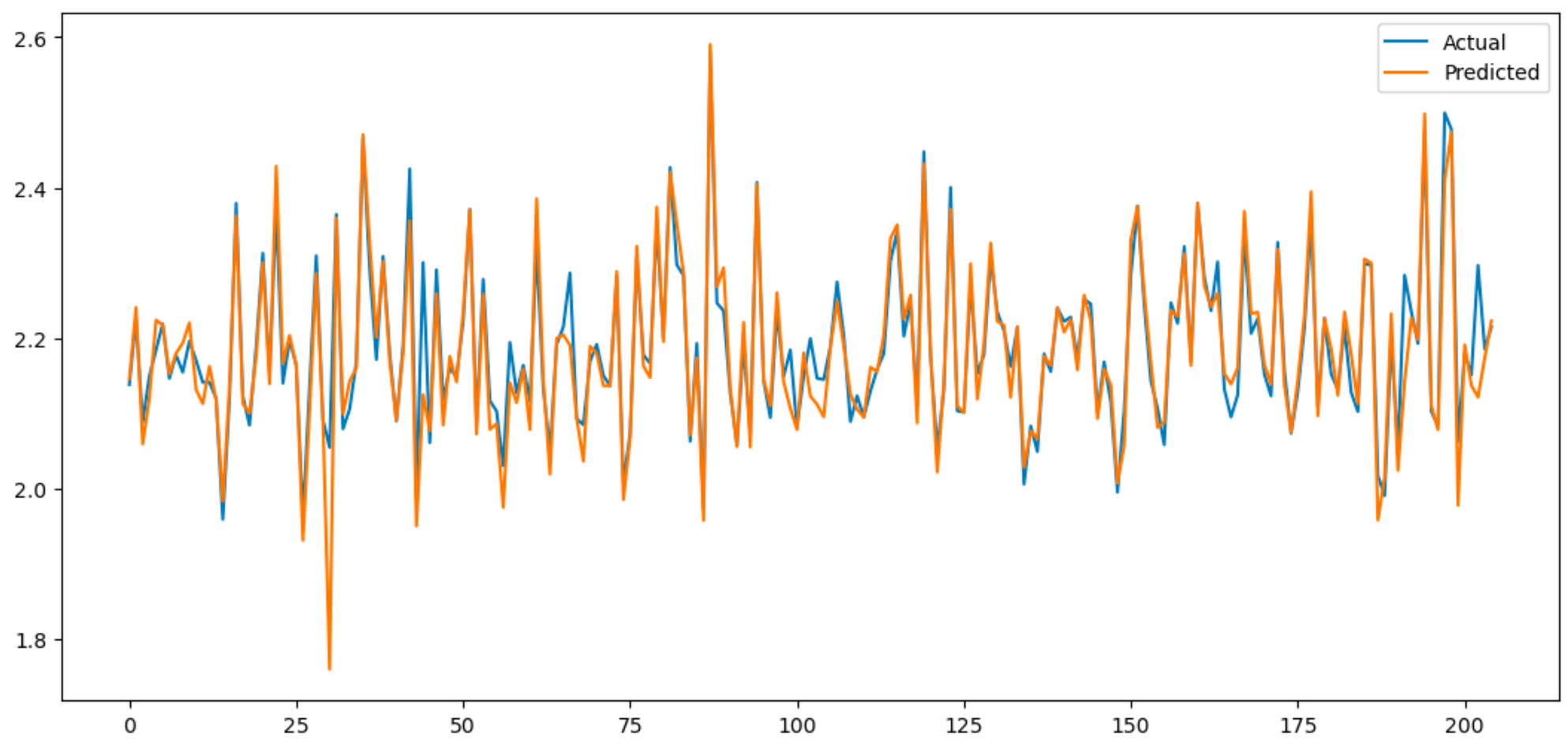
- We created an ML model which to predict the specific energy.
- Some columns were dropped in the process due to their high p-value.
- Rest of the columns are arranged in descending order of their coefficient which is also the order of their significance.

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c241 | R-squared: | 0.999 |
| Model: | OLS | Adj. R-squared: | 0.999 |
| Method: | Least Squares | F-statistic: | 9505. |
| Date: | Mon, 13 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 04:35:41 | Log-Likelihood: | 1921.1 |
| No. Observations: | 820 | AIC: | -3652. |
| Df Residuals: | 725 | BIC: | -3205. |
| Df Model: | 94 | | |
| Covariance Type: | nonrobust | | |

TASK 5.1

# PREDICTING SPECIFIC ENERGY



## TASK 5.1

# PREDICTING SPECIFIC ENERGY

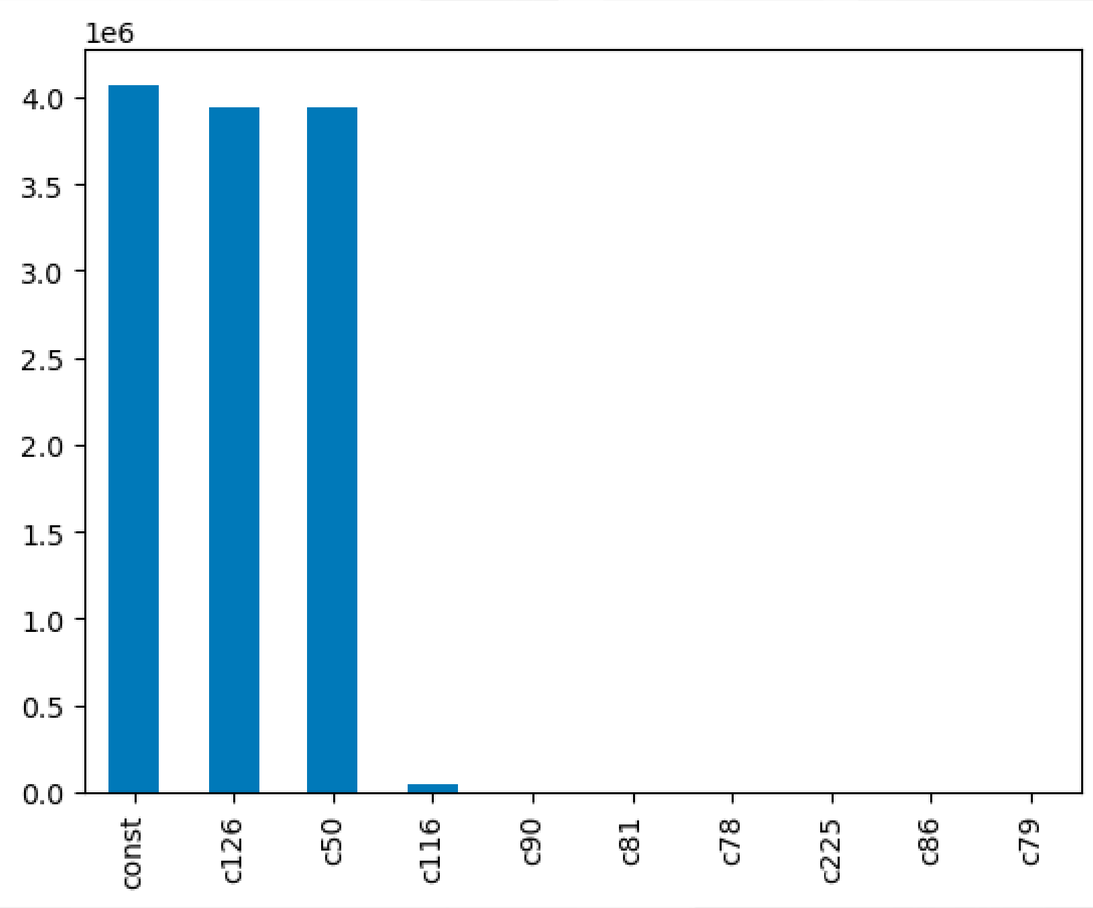
Number of columns dropped : 63

Number of significant columns : 101 + 1 constant

| | | |
|---|---|---|
| 1 | const | 4.069762e+06 |
| 2 | c126 | 3.939883e+06 |
| 3 | c50 | 3.939881e+06 |
| 4 | c116 | 4.016678e+04 |
| 5 | c90 | 2.549610e+02 |
| 6 | c81 | 2.282888e+02 |
| 7 | c78 | 1.893073e+02 |
| 8 | c225 | 1.848081e+02 |
| 9 | c86 | 1.804518e+02 |
| 10 | c79 | 1.075146e+02 |
| 11 | c123 | 8.728776e+01 |
| 12 | c132 | 8.475849e+01 |
| 13 | c98 | 2.753974e+01 |
| 14 | c99 | 2.224584e+01 |
| 15 | c94 | 2.026836e+01 |
| 16 | c201 | 1.888117e+01 |
| 17 | c224 | 1.867260e+01 |
| 18 | c39 | 1.639921e+01 |
| 19 | c92 | 1.297475e+01 |
| 20 | c91 | 9.762311e+00 |

Significant columns & their coefficient (Desc)

**Most Significant Column: c126**



TASK 5.2

# ONLY “PREDICTING” SPECIFIC ENERGY

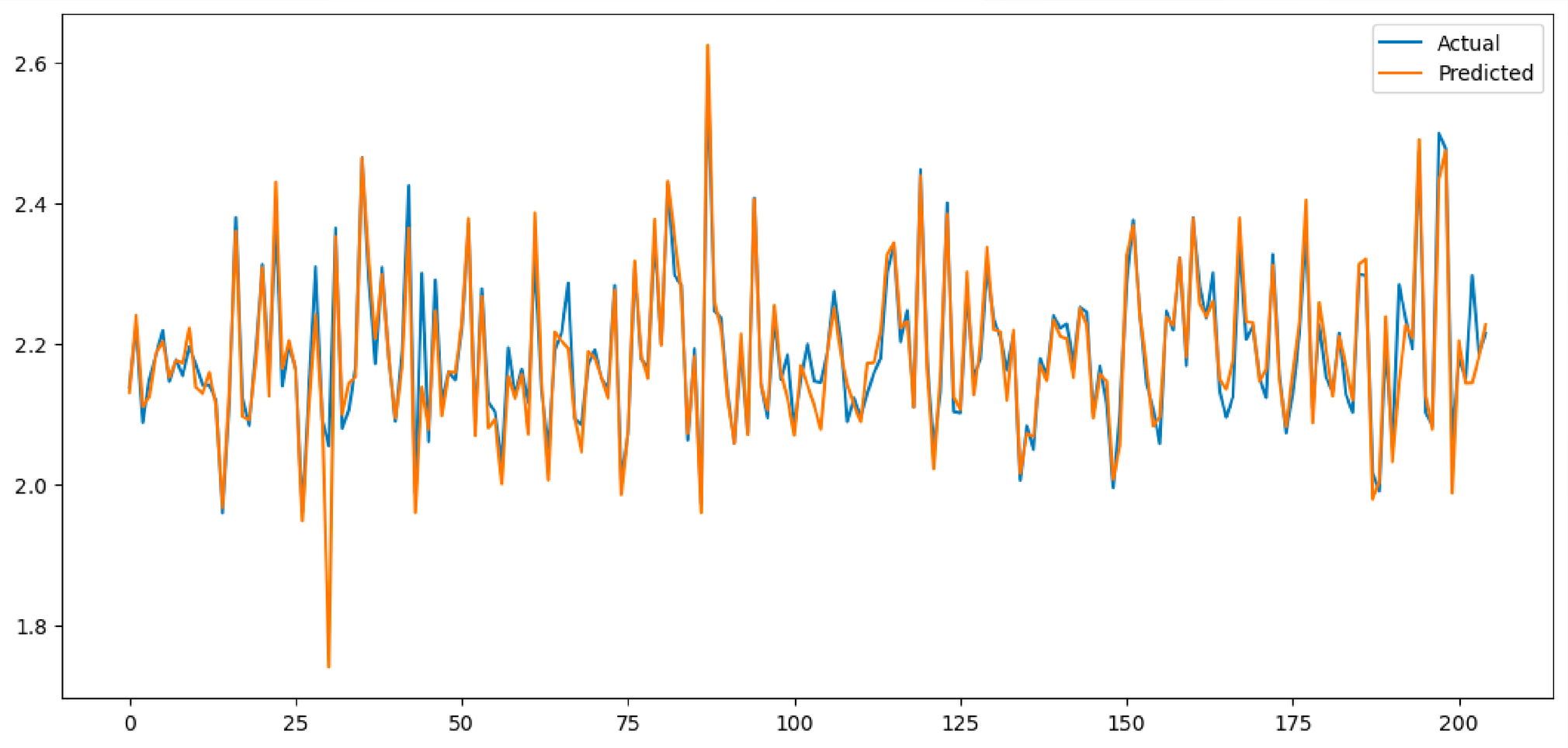
- We created another ML model which takes only the operating parameters as input.
- Again, like 5.1, we find the significance level of the columns.

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | c241 | R-squared: | 0.999 |
| Model: | OLS | Adj. R-squared: | 0.999 |
| Method: | Least Squares | F-statistic: | 9617. |
| Date: | Mon, 13 Nov 2023 | Prob (F-statistic): | 0.00 |
| Time: | 05:02:50 | Log-Likelihood: | 1895.4 |
| No. Observations: | 820 | AIC: | -3613. |
| Df Residuals: | 731 | BIC: | -3194. |
| Df Model: | 88 | | |
| Covariance Type: | nonrobust | | |

TASK 5.2

# ONLY “PREDICTING” SPECIFIC ENERGY



## TASK 5.2

# ONLY “PREDICTING” SPECIFIC ENERGY

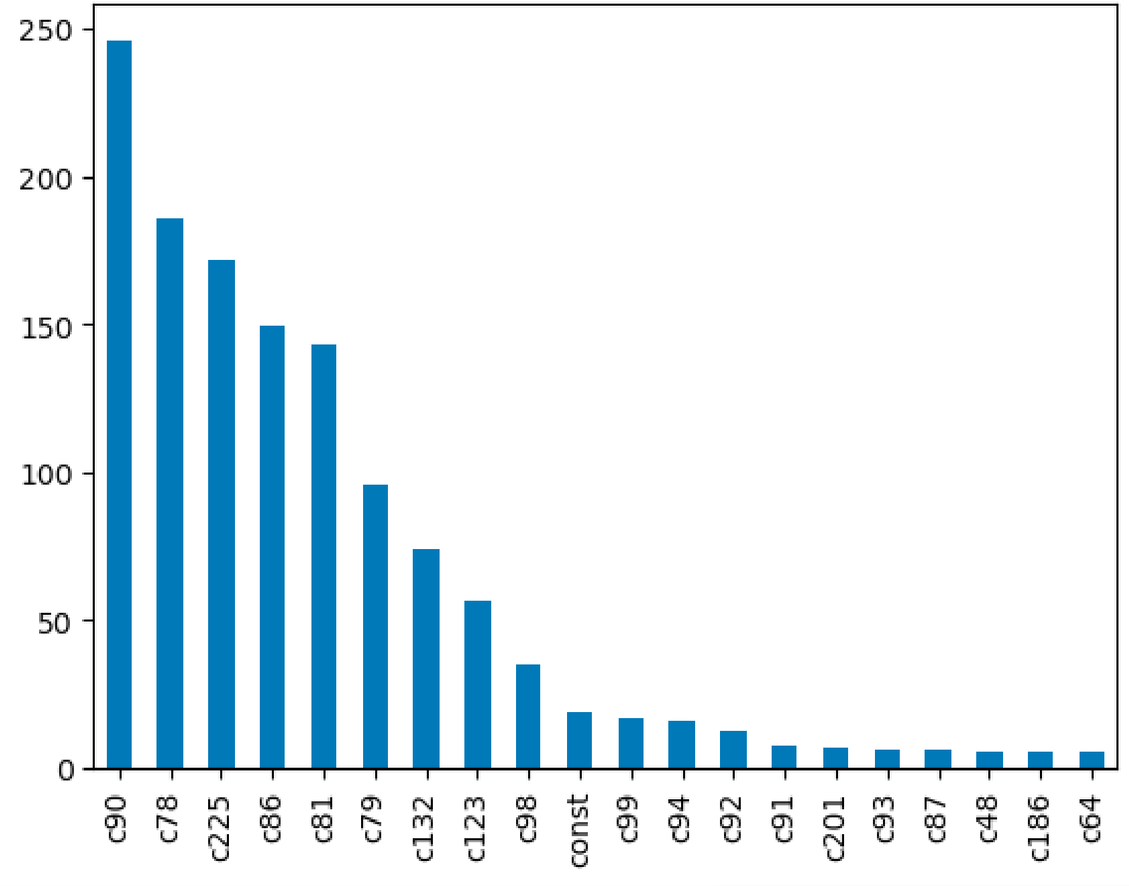
Number of columns dropped : 56

Number of significant columns : 90 + 1 constant

| | | |
|---|---|---|
| 1 | c90 | 246.073184 |
| 2 | c78 | 185.883616 |
| 3 | c225 | 171.639079 |
| 4 | c86 | 149.279650 |
| 5 | c81 | 143.258271 |
| 6 | c79 | 95.309498 |
| 7 | c132 | 73.779214 |
| 8 | c123 | 56.426831 |
| 9 | c98 | 34.705927 |
| 10 | const | 18.370343 |
| 11 | c99 | 16.584502 |
| 12 | c94 | 15.667384 |
| 13 | c92 | 12.429775 |
| 14 | c91 | 7.350902 |
| 15 | c201 | 6.817545 |
| 16 | c93 | 6.331665 |
| 17 | c87 | 5.963317 |
| 18 | c48 | 5.746224 |
| 19 | c186 | 5.484521 |
| 20 | c64 | 5.383652 |

Significant columns & their coefficient (Desc)

**Most Significant Column: c90**



# OUR ACHIEVEMENTS

**Cleaned the data and made it usable for further analysis**

- Trained an ML model to predict vibrations from the input parameters.
- Sucessfully created an automated mechanism which fine tunes the input parameters to keep the vibrations of the critical equipments in safe-moderate ranges.

- Trained ML models to predict the Specific Energy from the input parameters (One for all parameters, One only for operating parameters)
- Found the significance of different parameters on the specific energy


# THANK YOU