Prediction and Analysis of Thermal Errors for Efficient CNC Machining

MEB1213 - Ashutosh Garg
MEB1214 - Ashwin Goyal
MEB1215 - Ayush Agarwal

Thermal error and temperature data extracted from graphs of the paper:

Thermal error optimization modeling and real-time compensation on a CNC turning center

JOURNAL OF MATERIALS PROCESSING TECHNOLOGY 207 (2008) 172–179

ELSEVIER

journal homepage: www.elsevier.com/locate/jmatprotec

Thermal error optimization modeling and real-time compensation on a CNC turning center

Wu Hao, Zhang Hongtao, Guo Qianjian, Wang Xiushan, Yang Jianguo*

School of Mechanical Engineering, Shanghai Jiaotong University, Shanghai 200240, PR China

ARTICLE INFO

Article history:
Received 22 May 2007
Received in revised form
24 October 2007
Accepted 15 December 2007

Keywords:
Thermal error
Optimization modeling
Genetic algorithm
Artificial neural networks
NC machine tool

ABSTRACT

Thermal errors are the largest contributor to the dimensional errors of a workpiece in precision machining. The error compensation technique is an effective way of reducing thermal errors. Accurate modeling of errors is a key part of error compensation. The thermal errors of a machine tool can be treated as the superposition of a series of thermal error modes. In this paper, five key temperature points of a turning center were obtained based on the thermal error mode analysis. A thermal error model based on the five key temperature points was proposed by using genetic algorithm-based back propagation neural network (GA-BPN). The GA-BPN method improves the accuracy and reduces computational cost for the prediction of thermal deformation in the turning center. A thermal error real-time compensation system was developed based on the proposed model. An experiment was carried out to verify the performance of the compensation system. The experimental results show that the diameter error of the workpiece reduced from about 27–10 μm after implementation of the compensation.

1. Introduction

The inherent inaccuracy of machine tools is a major contributor to workpiece errors. Among many various sources of machine tool errors, thermal errors are the largest contributor, accounting for as much as 70% of workpiece errors in precision machining (Weck et al., 1995). Researchers have considered many ways of reducing thermal errors, including the thermally symmetric design of a structure, separation of the heat sources from the main body of a machine tool, installation of a cooling unit, and so on. However, the manufacturing costs associated with the above-mentioned approaches are usually very high. In addition, there are many physical limitations in implementing process, which cannot be overcome solely by design techniques. As a result, error compensation technique used to improve machine accuracy cost-effectively has received significant attention in recent years (Yang et al., 1996a).

Accurate modeling of errors is a key part of error compensation. The thermal errors of a machine tool origin from the non-linear and time-varying thermal deformations caused by the non-uniform temperature variations in the machine structure. The temperature variations are related to the heat source location, heat source intensity, thermal resistance coefficient and the machine system configuration. Therefore, the thermal error model is usually achieved by non-linear empirical modeling approaches which correlate machine thermal errors to temperature measurements of a machine. In recent years, it has been shown that thermal error map of a machine tool can be successfully approximated by empirical modeling approaches such as multiple regression analysis techniques (Yang et al., 1996b, 1999, 2002; Lee and Yang, 2002), types of artificial neural networks (Yang et al., 1996c; Yang and Lee, 1998; Mize and Ziegert, 2000; Lee et al., 2003; Yang and Ni, 2005), grey system theory (Wang et al., 1998), genetic algorithm (Choi and Lee, 2002), rigid body kinematics (Okafor

* Corresponding author. Tel.: +86 13817082992.
E-mail addresses: wuhaoer@sjtu.edu.cn (W. Hao), jgyang@sjtu.edu.cn (Y. Jianguo).
0924-0136/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.jmatprotec.2007.12.067

What are we going to Discuss in next 8-9 minutes?

- The Motive
- Different sensor positions
- The Code: Different Models Proposed
- Comparisons & Results
- Uses & Applications

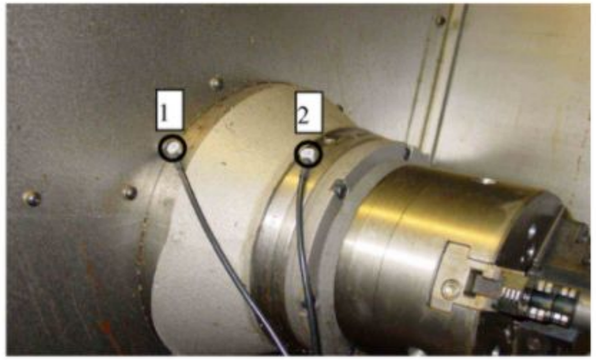
The Motive

- Minimizing Errors
- Data visualization
- Parameters and sensors
- Machine Learning Model Analysis

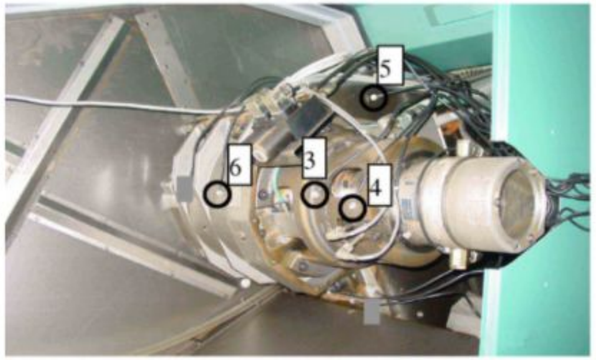
Temperature Sensor Positions & Effects of Temp. Rise

- Friction- Cause of temperature Rise
- Bearing Expansion
- Ball Screw Expansion
- Wrapping Effect of Base

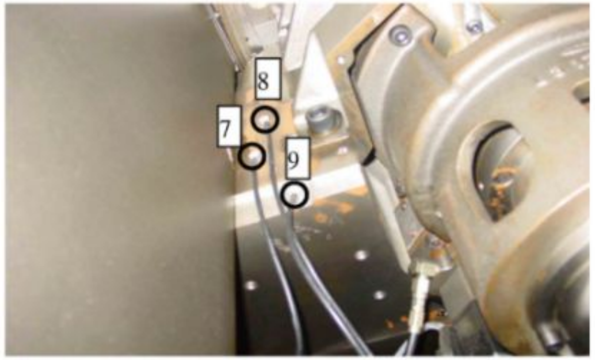
(a) The tested turning center



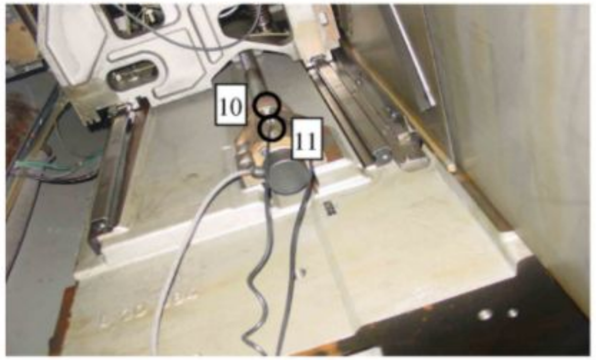
(b) Thermistors on the spindle fore-end



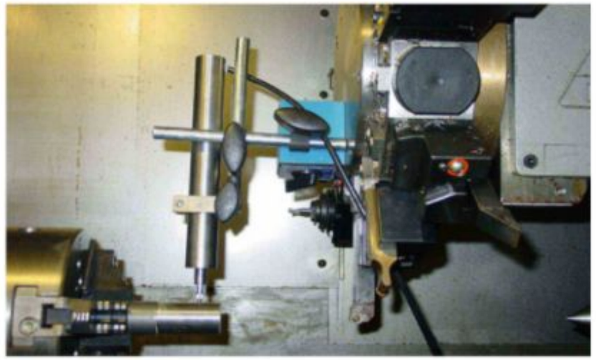
(c) Thermistors on the spindle rear-end



(d) Thermistors on the headstock



(e) Thermistors on the X-axis lead screw



(f) Displacement sensor

Inside the Code

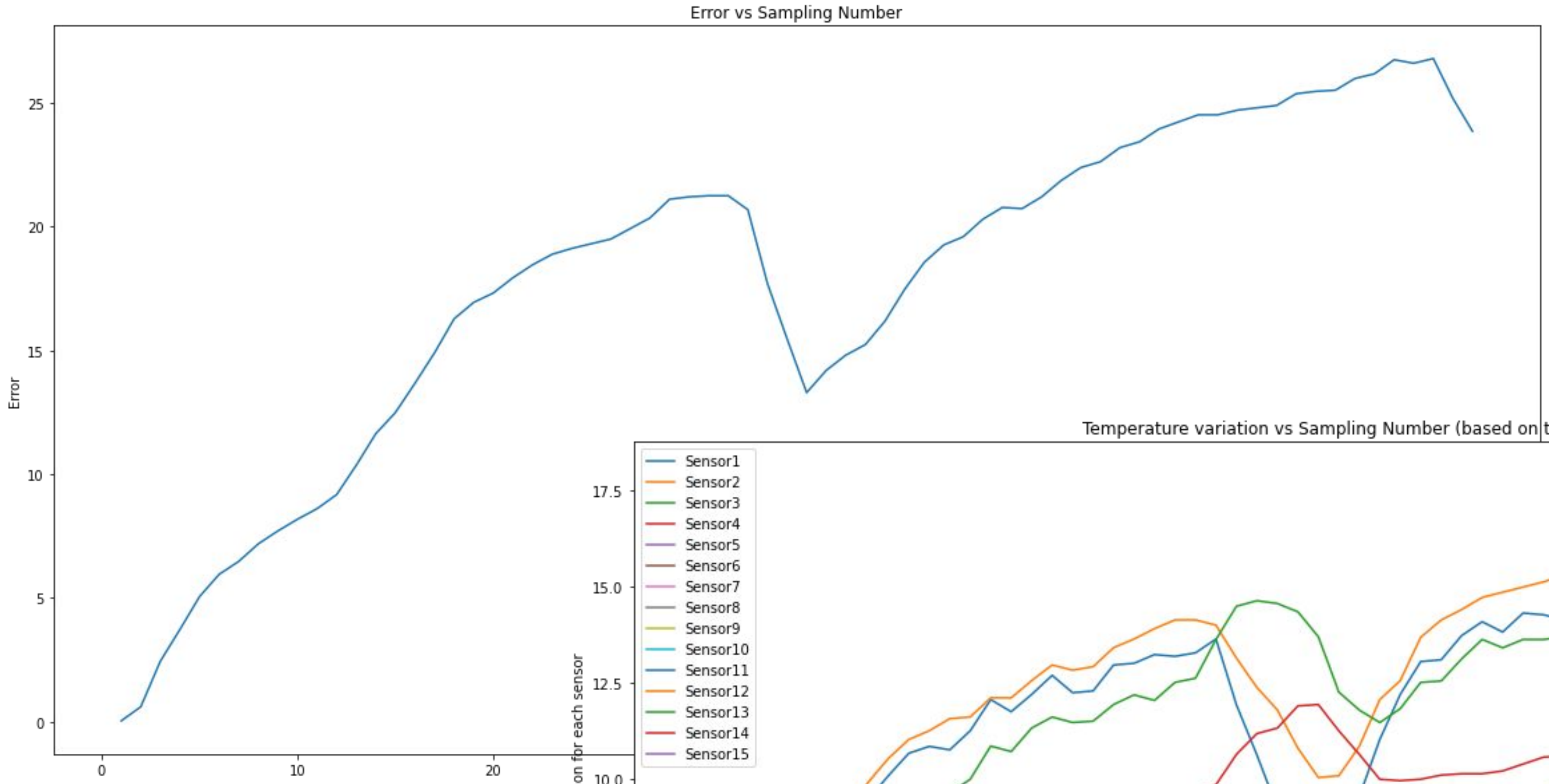
Data Visualization

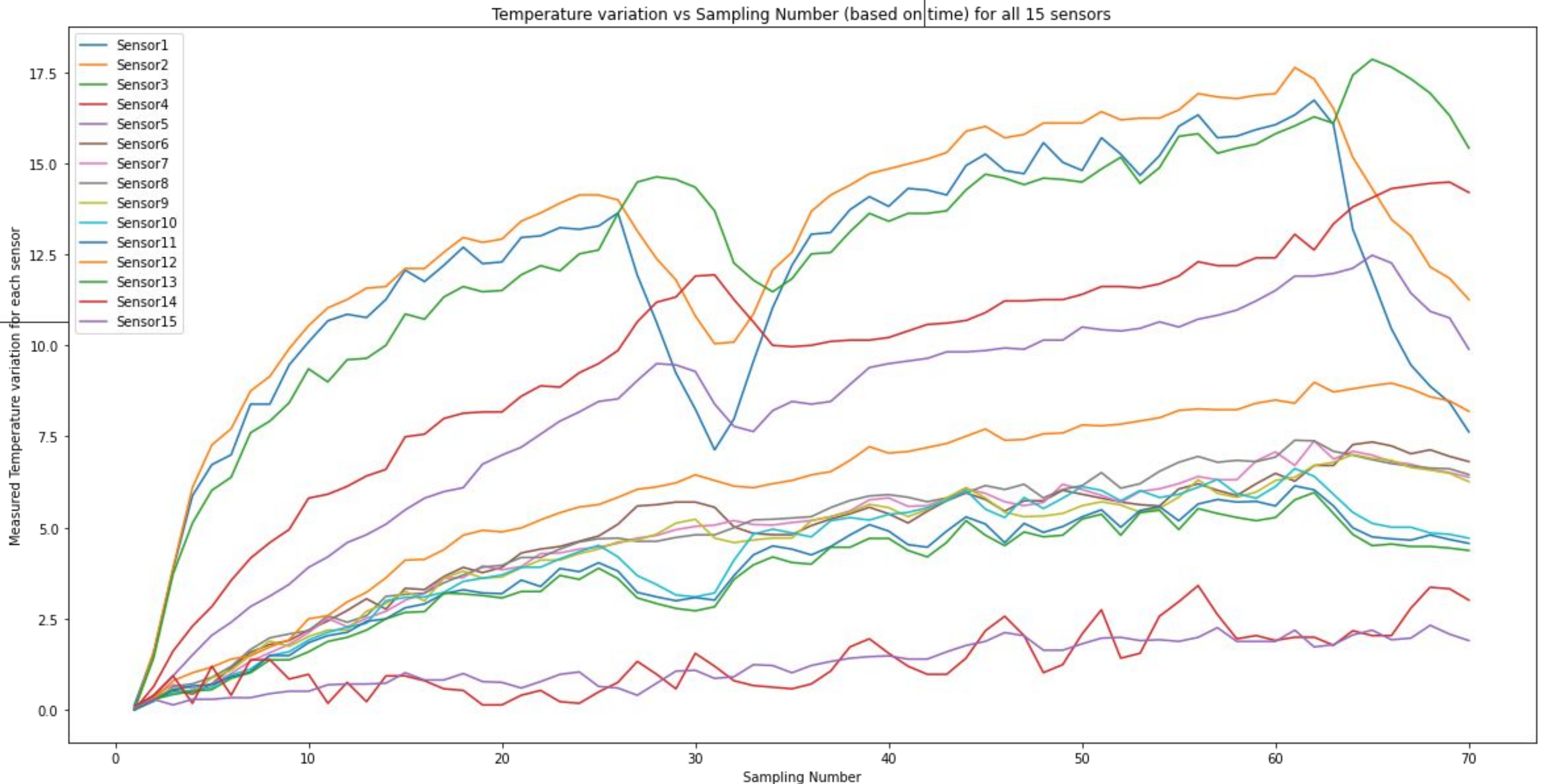
Choosing the Right Sensors

Regression Using Different Approaches

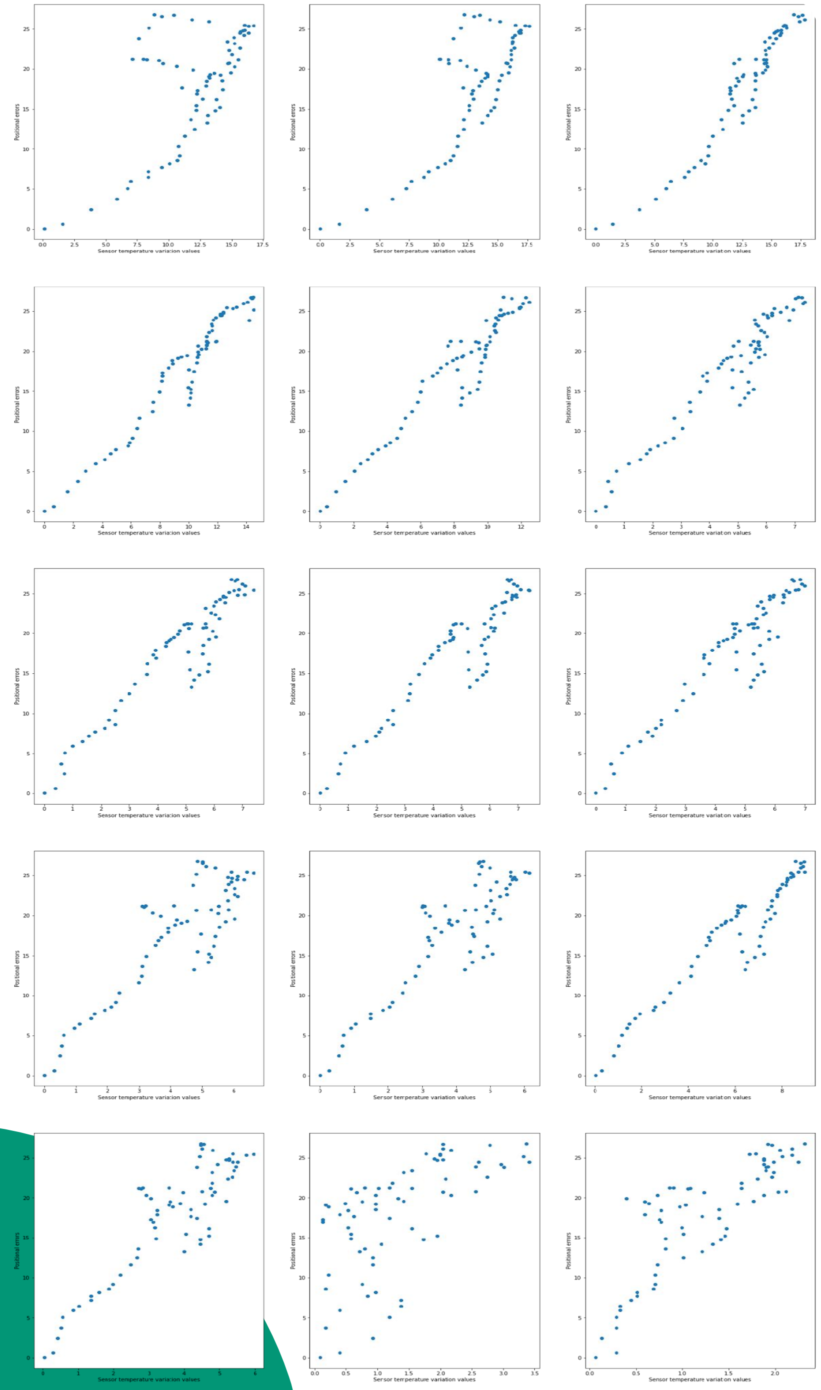
Comparison of Results

DATA VISUALIZATION





Click to add text



Sensor temperature variation values vs **Positional errors**

3 key learnings:

- For Sensors 3-8 it is almost linear and is uniformly spread out
- For other sensors, there is some non-uniformity and variations are more complex than Linear
- Last 2 sensors almost convey no major information

Choosing the Right Sensors

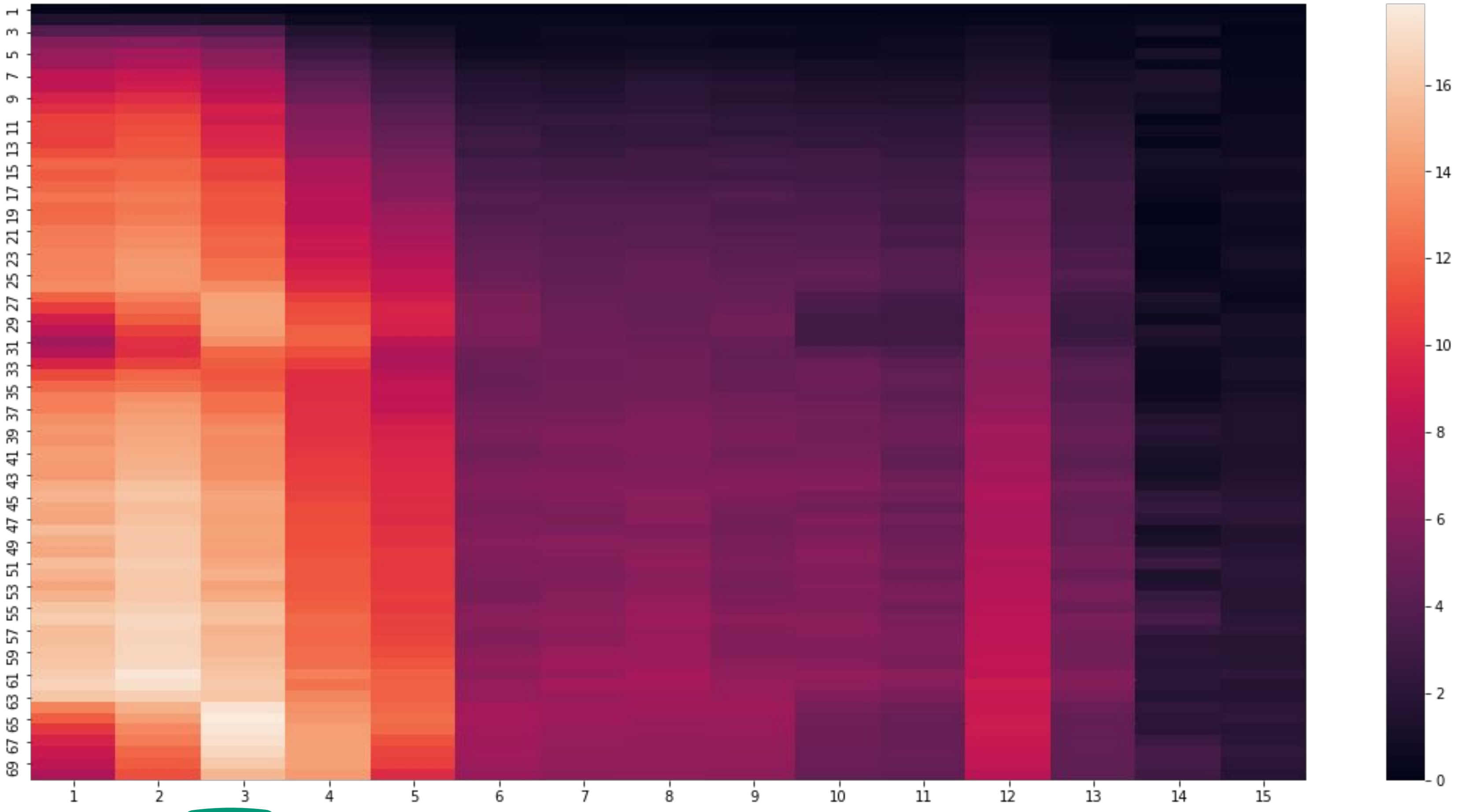
1

Insights from Heatmap

2

Clustering based on correlation

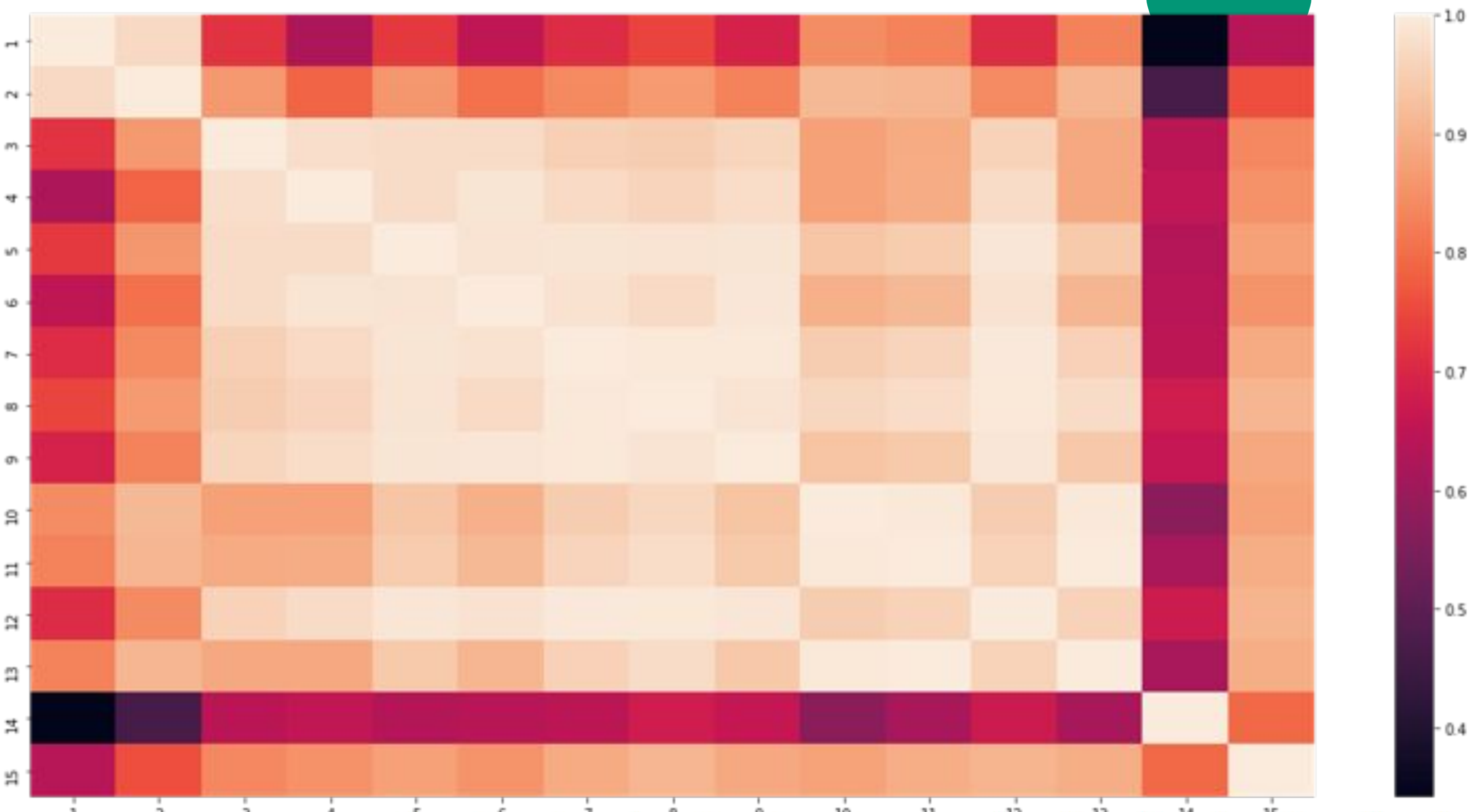
1. Insights from Heatmap and sensor positions



- **1 to 5:** High Variance (More Information)
- Others do not vary much- they have low temperature variations
- Sensor positions from 1 to 5 along with high variance, are also directly responsible for holding the work.

2. Clustering Based on Correlation

Correlation Matrix + **Code** + **Heatmap** confirm our results



```
[131] import scipy.cluster.hierarchy as sci

      corr = df1.corr().values

      pd = sci.distance.pdist(corr)
      linkage = sci.linkage(pd, method='complete')
      id = sci.fcluster(linkage, 0.2 * pd.max(), 'distance')
      id

      array([4, 5, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 1, 6, 3], dtype=int32)
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.000000	0.968485	0.718261	0.622147	0.730066	0.651168	0.708134	0.744505	0.691249	0.844082	0.829502	0.708757	0.829957	0.341901	0.639990
2	0.968485	1.000000	0.861207	0.785907	0.860710	0.802997	0.838351	0.864254	0.828655	0.915146	0.909225	0.840624	0.908238	0.467531	0.758112
3	0.718261	0.861207	1.000000	0.978032	0.972938	0.973074	0.952374	0.947979	0.962769	0.876563	0.891771	0.957582	0.887613	0.643839	0.836836
4	0.622147	0.785907	0.978032	1.000000	0.973335	0.988922	0.970407	0.960432	0.975980	0.875695	0.893531	0.972351	0.888744	0.654988	0.851088
5	0.730066	0.860710	0.972938	0.973335	1.000000	0.985781	0.988133	0.984911	0.987551	0.934868	0.944721	0.990955	0.941331	0.634639	0.875725
6	0.651168	0.802997	0.973074	0.988922	0.985781	1.000000	0.983729	0.970820	0.990581	0.900009	0.912870	0.983701	0.909085	0.640455	0.854883
7	0.708134	0.838351	0.952374	0.970407	0.988133	0.983729	1.000000	0.993567	0.992391	0.947837	0.959015	0.995525	0.955843	0.649114	0.890956
8	0.744505	0.864254	0.947979	0.960432	0.984911	0.970820	0.993567	1.000000	0.986410	0.965113	0.975236	0.993409	0.972336	0.677402	0.909938
9	0.691249	0.828655	0.962769	0.975980	0.987551	0.990581	0.992391	0.986410	1.000000	0.931013	0.942613	0.991880	0.939764	0.658734	0.885860
10	0.844082	0.915146	0.876563	0.875695	0.934868	0.900009	0.947837	0.965113	0.931013	1.000000	0.992908	0.947909	0.993236	0.575221	0.877444
11	0.829502	0.909225	0.891771	0.893531	0.944721	0.912870	0.959015	0.975236	0.942613	0.992908	1.000000	0.958766	0.997821	0.615406	0.895089
12	0.708757	0.840624	0.957582	0.972351	0.990955	0.983701	0.995525	0.993409	0.991880	0.947909	0.958766	1.000000	0.956370	0.672977	0.906602
13	0.829957	0.908238	0.887613	0.888744	0.941331	0.909085	0.955843	0.972336	0.939764	0.993236	0.997821	0.956370	1.000000	0.616067	0.894744
14	0.341901	0.467531	0.643839	0.654988	0.634639	0.640455	0.649114	0.677402	0.658734	0.575221	0.615406	0.672977	0.616067	1.000000	0.792518
15	0.639990	0.758112	0.836836	0.851088	0.875725	0.854883	0.890956	0.909938	0.885860	0.877444	0.895089	0.906602	0.894744	0.792518	1.000000

Thermal Error Prediction Approaches:

Linear Regression

PCA

Neural Networks

Polynomial Regression

Linear Regression

```
[11] from sklearn import datasets, linear_model
     from sklearn.metrics import mean_squared_error,confusion_matrix

     from sklearn.linear_model import LinearRegression

     reg = linear_model.LinearRegression()

     # Train the model using the training sets
     reg.fit(X_train, y_train)

     # Make predictions using the testing set
     y_pred = reg.predict(X_test)
     y_pred

     array([28.65677213, 17.47649304, 18.15863582, 28.98336201, 25.05728277,
            25.62511935, 27.37608329, 16.95618079, 25.92340552, 22.59560472,
            18.35944864, 23.13065671,  9.9362597 , 13.31207123, 18.3647796 ,
            19.16731123, 11.40686787, 26.57640344, 15.74711226, -1.68657532,
            22.87553239])

[12] y_test

     array([26.149, 17.685, 20.664, 26.716, 25.345, 25.44 , 26.574, 14.8  ,
            25.156, 20.333, 19.292, 21.184,  8.606, 12.483, 16.172, 19.576,
            11.632, 25.96 , 17.306,  0.047, 22.602])

[13] print('Coefficients: \n', reg.coef_)
     # mean squared error
     print('Mean squared error: %.2f' % mean_squared_error(y_test, y_pred))

     Coefficients:
      [ 0.53692302 -1.48916046  1.77198791  0.73190075  1.86866492  0.05282001
       0.23555601 -0.8638938  -4.15419536 -0.52287336 -1.22174307  0.29677728
       4.00336151 -0.39301756  0.68355898]
     Mean squared error: 2.23
```

- Minimum MSE through this approach- data has a linear setting
- Ran more than once on particular sensors

PCA

```
[19] #PCA
     from sklearn.decomposition import PCA

     comp=100
     pca=PCA(n_components=5)
     pca.fit(X_train)

     X_train_pca=pca.transform(X_train)
     X_test_pca=pca.transform(X_test)
     #X_train_pca.shape
     X_train_pca.shape

     (49, 5)
```

```
eigenvalues = pca.explained_variance_
eigenvalues

array([80.5449243 ,  6.51457542,  0.99883578,  0.509529  ,  0.1246482 ])
```

```
[22] cov_matrix = pca
     cov_matrix.fit(X_train)
     variance = cov_matrix.explained_variance_ratio_
     vars=np.cumsum(np.round(cov_matrix.explained_variance_ratio_, decimals=3)*100)
     vars #cumulative sum of variance
     #Clearly first 5 features give almost 100% variance
     #SO WE CAN SAFELY TAKE ONLY FIRST 5 FEATURES(Sensors)

     array([90.6, 97.9, 99. , 99.6, 99.7])
```

```
[23] #Now using Linear regrgession on the PCA components
```

```
[24] reg = linear_model.LinearRegression()

     # Train the model using the training sets
     reg.fit(X_train_pca, y_train)

     # Make predictions using the testing set
     y_pred_pca = reg.predict(X_test_pca)
     y_pred_pca
```

```
y_test

array([26.149, 17.685, 20.664, 26.716, 25.345, 25.44 , 26.574, 14.8  ,
       25.156, 20.333, 19.292, 21.184,  8.606, 12.483, 16.172, 19.576,
       11.632, 25.96 , 17.306,  0.047, 22.602])
```

```
[26] print('Mean squared error: %.2f' % mean_squared_error(y_test, y_pred_pca))

     Mean squared error: 3.07
```

- Found out new, more informative features (Number of components=5)
- Explained Variance- first 5 eigen vectors taken for maximum information
- These collectively express 99.8% variance

Neural Network

```
[148] from keras import Sequential
      from keras.layers import Dense

      model = Sequential()
      model.add(Dense(20, activation='relu',kernel_initializer='normal', input_dim=15))
      model.add(Dense(5, activation='relu',kernel_initializer='normal'))
      model.add(Dense(1, kernel_initializer='normal'))

      model.compile(optimizer ='adam',loss='mean_squared_error')

      #X_train = np.asarray(X_train)
      #y_train = np.asarray(y_train)
      #X_val = np.asarray(X_val)
      #y_val = np.asarray(y_val)

      model.fit(X_train,y_train, batch_size=10, epochs=100)
```

```
[150] y_test

      array([25.345, 16.928,  6.478, 20.333, 25.487, 21.184, 22.366, 20.285,
              5.06 ,  8.606, 16.266, 21.089, 18.867,  2.459, 19.292, 23.17 ,
             20.664, 24.21 , 26.716, 25.44 , 14.895])
```

```
[151] print('Mean squared error: %.2f' % mean_squared_error(y_test, y_pred_nn))

      Mean squared error: 3.86
```

- Neural Network for regression
  - Output layer has no activation function
  - Other layers use ReLU function
  - 2 hidden layers
  - Epochs 100
- Objective function: MSE
- NN ran thrice- for different variations

Polynomial Regression

```
import timeit
start = timeit.default_timer()

from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=3)
X_poly = poly_reg.fit_transform(X_train)
pol_reg = LinearRegression()
pol_reg.fit(X_poly, y_train)

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)


stop = timeit.default_timer()
print('Time: ', stop - start)
```

```
Time:  0.01660974299738882
```

```
[118] y_pred_lin=lin_reg.predict(X_test)
      # Predicting a new result with Polymonial Regression
      y_pred_poly=pol_reg.predict(poly_reg.fit_transform(X_test))
      y_pred_poly
```

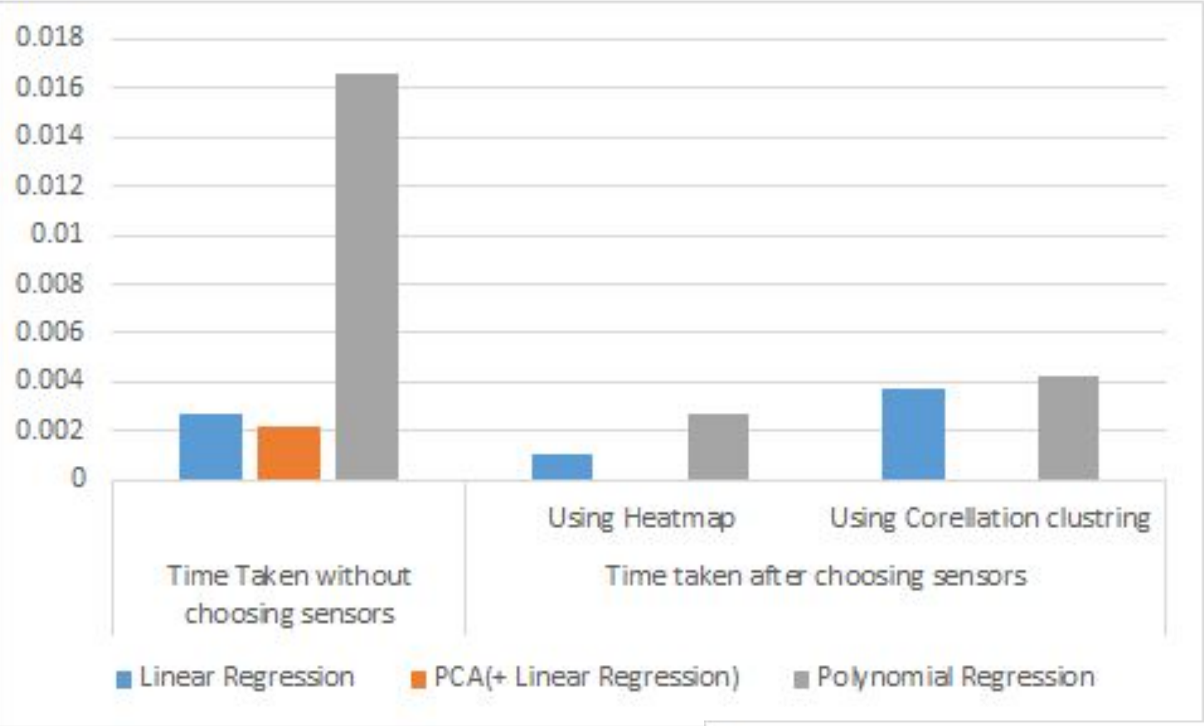
```
array([ 2.2834229 , 17.15119315, 23.20856834, 16.26975135, 16.45771219,
       12.07560442, 14.14706895, 20.26814796, 30.82674774, 15.85532802,
       23.61279125,  4.42657597, 26.68715072, 22.68009587, 30.06993318,
       20.42438251, 29.31923635, 15.89430274, 18.34042432, 16.49234431,
       23.37244115])
```

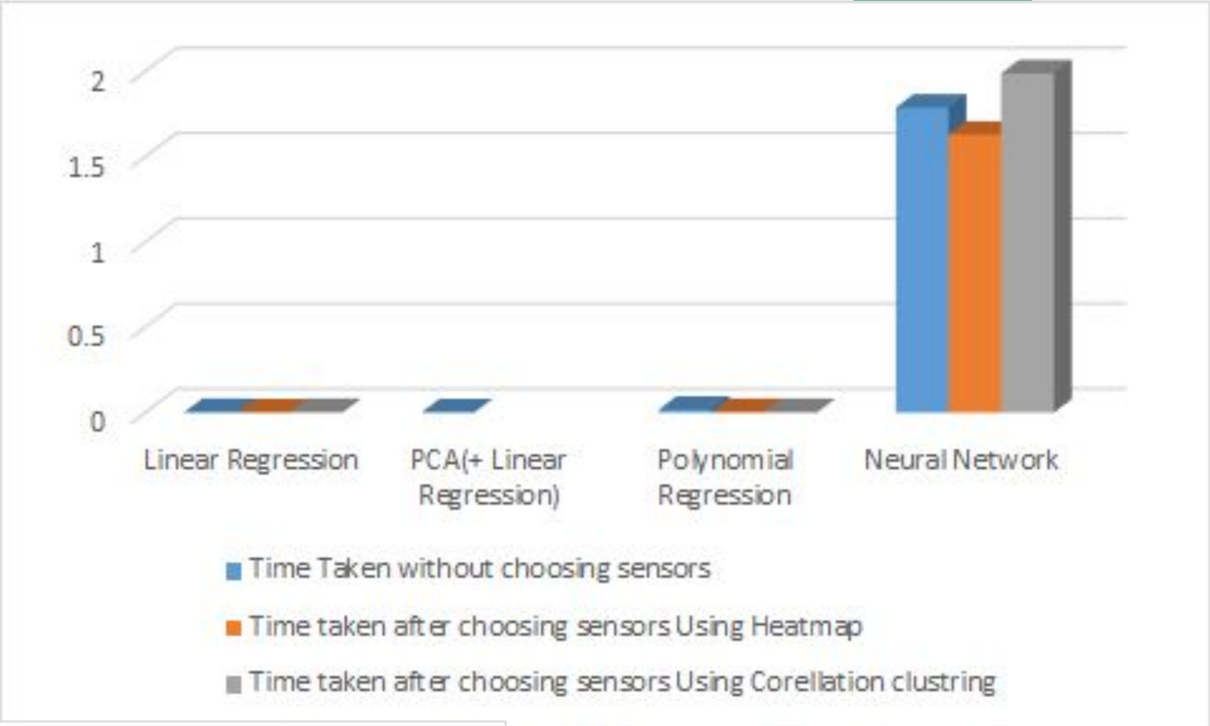
- On increasing the degree to 4 or more, it was overfitting
- Data points are limited, hence there's a high possibility of overfitting

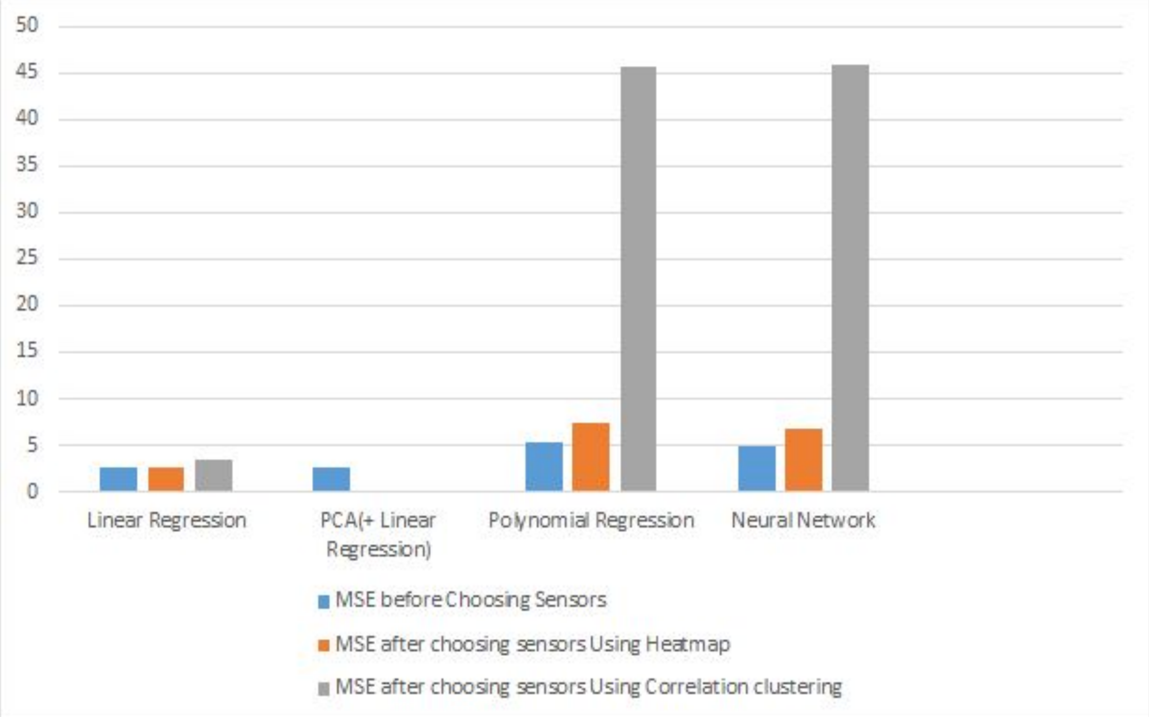
Comparisons

Regression Name	MSE Before choosing sensors	MSE after choosing sensors		Time taken without choosing sensors	Time taken after choosing sensors.	
		Using heatmap	Using correlation clustering		Using heatmap	Using correlation clustering
Linear Regression	2.62	2.57	3.47	0.00266	0.00107	0.00378
PCA(+ linear regression)	2.67	N/A	N/A	0.00220 (+0.00097=0.00317)	N/A	N/A
Neural Network	4.81	6.69	45.96	1.78879	1.62970	1.98920
Polynomial Regression	5.32	7.37	45.70	0.01660	0.00270	0.00423

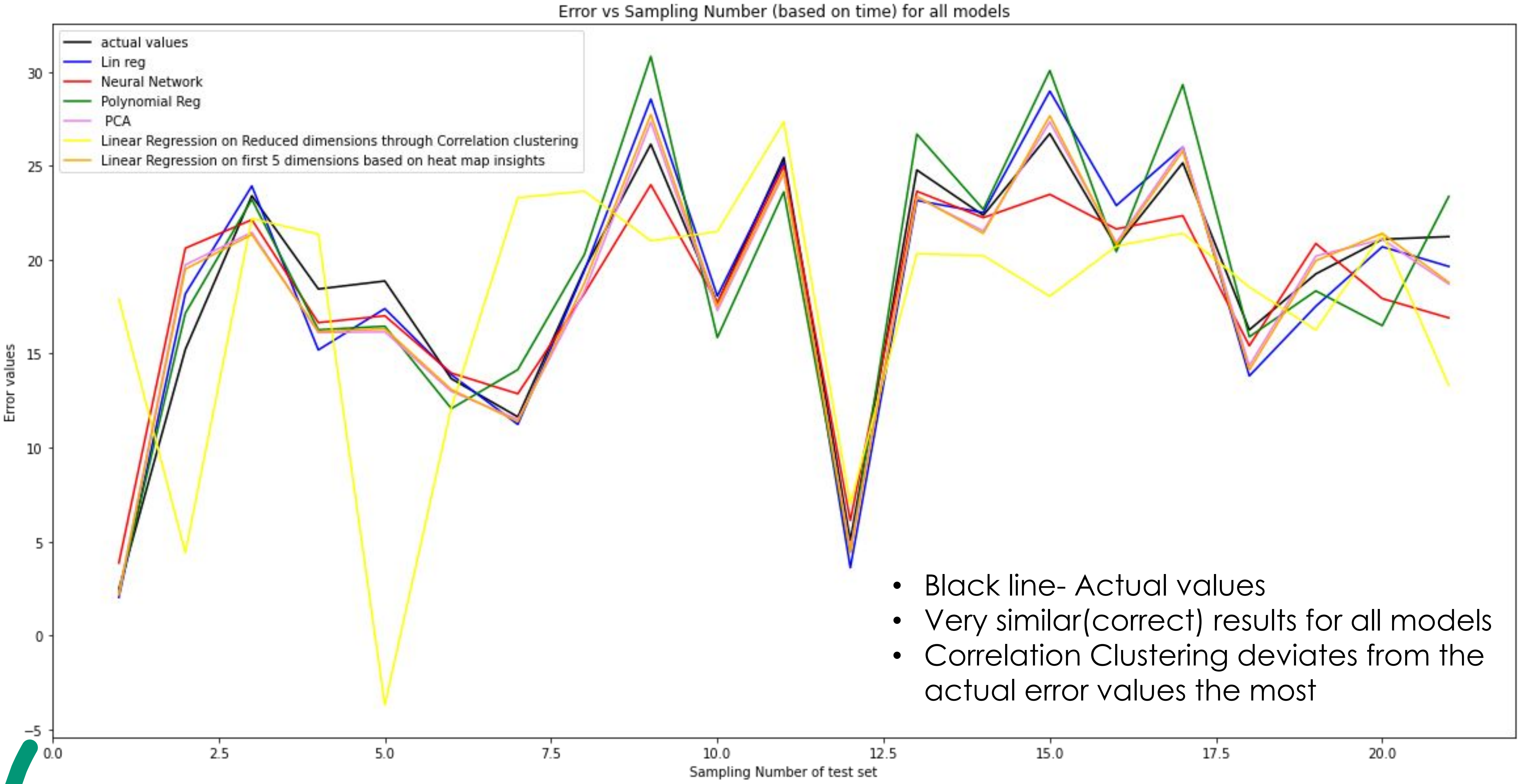
- The minimum time is for linear regression with sensors chosen
- With correlation clustering, the errors are getting increased. Mainly this could be because of non-relevance to the thermal error data. The correlation among the features does not exactly convey whether they carry similar information(temp variation) or not


0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
Using Heatmap
Using Corellation clustring
Time Taken without choosing sensors
Time taken after choosing sensors
Linear Regression
PCA(+ Linear Regression)
Polynomial Regression


2
1.5
1
0.5
0
Linear Regression
PCA(+ Linear Regression)
Polynomial Regression
Neural Network
Time Taken without choosing sensors
Time taken after choosing sensors Using Heatmap
Time taken after choosing sensors Using Corellation clustring


50
45
40
35
30
25
20
15
10
5
0
Linear Regression
PCA(+ Linear Regression)
Polynomial Regression
Neural Network
MSE before Choosing Sensors
MSE after choosing sensors Using Heatmap
MSE after choosing sensors Using Correlation clustering

Error visualization in different models

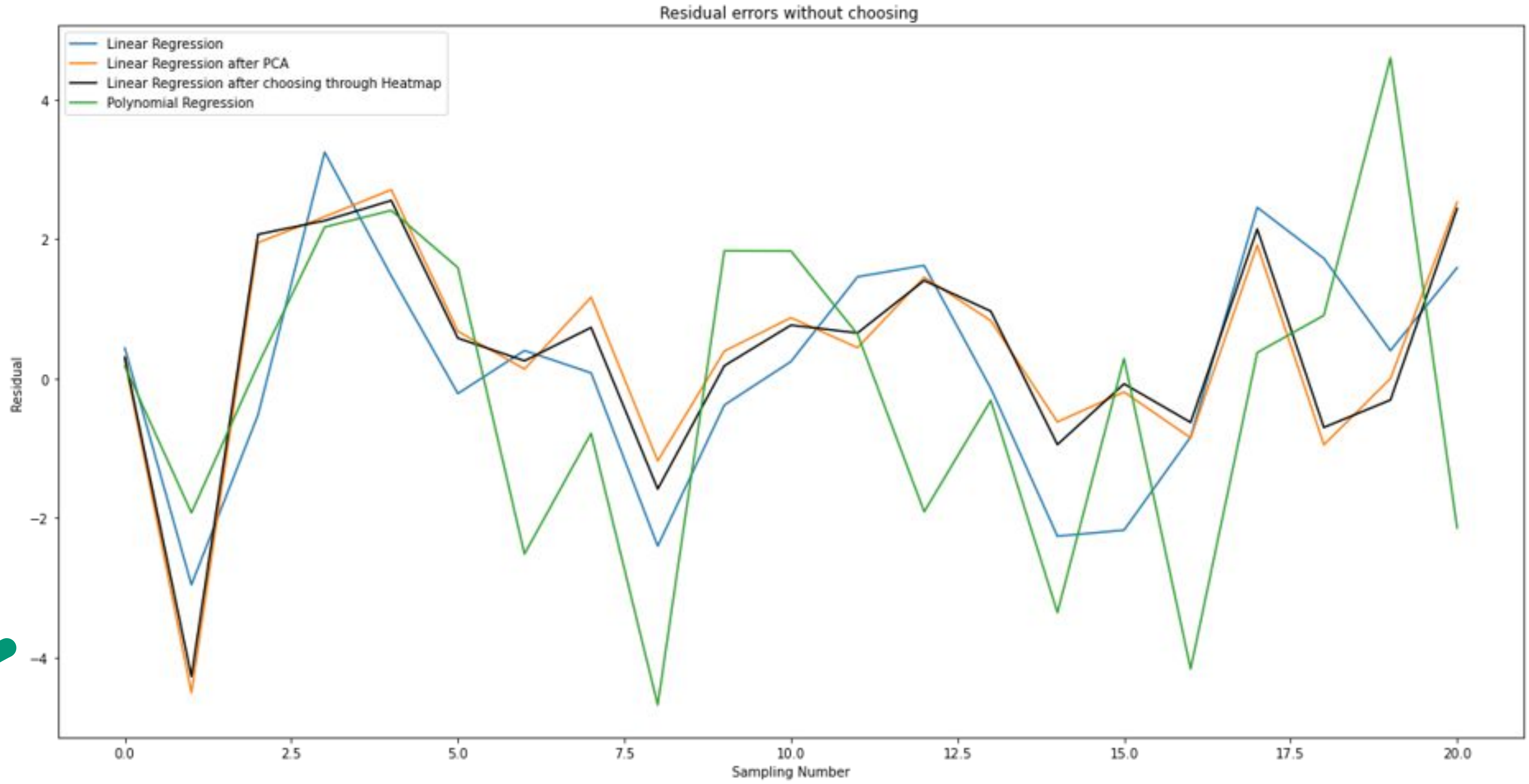


- Black line- Actual values
- Very similar(correct) results for all models
- Correlation Clustering deviates from the actual error values the most

Application:

Correction in Positioning during live operation

- Linear Regression with reduced sensors can be used for this



Thank You All for Listening!!
We are open for Questions Now