Predicting Performance of Integrated Circuits using Machine Learning

Nikhil Gupta (guptan@smu.edu), Max Moro (mmoro@smu.edu), Joanna Duran (joannad@mail.smu.edu)

Abstract – Semiconductor manufacturing is a variable process and outcomes depend on several factors. To meet target specifications, some parameters are controlleds by design engineers. However, many parameters are beyond human control (e.g. process variation). The output variables that are measured after the manufacturing process is complete must fall within a specified range of values (target specification). Variation in the manufacturing process may lead to issues if the outputs are outside the minimum or maximum value of these specifications. Through this work, we aim to build a model that can be used to predict the performance of an integrated circuit. This model could be used to preemptively take actions to prevent specification violation after manufacturing.

Index Terms – Semiconductors, Integrated Circuits, Predictive Modeling, Machine Learning, Feature Engineering, Linear Regression, Principal Component Analysis (PCA), LASSO, LARS.

I. Introduction

IN semiconductor manufacturing environments, the question that is often asked is "Can we predict the performance before the device is manufactured and preemptively make changes when the output is expected to be outside the desired range?" The answer is yes and the current practice is to use electrical simulation (including Monte Carlo runs) to identify performance limits. However, this is very resource and time intensive as each electrical simulation can take several hours to run.

Figure 1 shows an example specification sheet for an integrated circuit. Each row is a single output with its respective minimum, typical and maximum measured values. Note however that not all values are populated. This may be due to several reasons including but not limited to time and cost constraints to measure this in hardware.

The objective for this project was to build an accurate model that could be used to predict the performance (min, typical, max values) of an integrated circuit. This model could be useful to preemptively take action to make sure the measured output is within specification limits after manufacturing. In other cases, this model could also be used to predict the limits in cases where it is time and cost prohibitive to measure this on hardware. A target accuracy of ±10% was desired from this model, but ±15% was also acceptable if these occurrences were rare.

T_J = –40°C to 150°C, VIN = 4.5 V to 17 V, PVIN = 1.6 V to 17 V (unless otherwise noted)

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
SUPPLY VOLTAGE (VIN AND PVIN PINS)					
PVIN operating input voltage		1.6		17	V
VIN operating input voltage		4.5		17	V
VIN internal UVLO threshold	VIN rising		4	4.5	V
VIN internal UVLO hysteresis			150		mV
VIN shutdown supply Current	EN = 0 V		2	5	µA
VIN operating—nonswitching supply current	VSENSE = 810 mV		600	800	µA
ENABLE AND UVLO (EN PIN)					
Enable threshold	Rising		1.21	1.26	V
Enable threshold	Falling	1.10	1.17		V
Input current	EN = 1.1 V		1.15		µA
Hysteresis current	EN = 1.3 V		3.4		µA
VOLTAGE REFERENCE					
Voltage reference	0 A ≤ I_{OUT} ≤ 6 A	0.792	0.8	0.808	V
MOSFET					
High-side switch resistance	BOOT-PH = 3 V		32	60	mΩ
High-side switch resistance(1)	BOOT-PH = 6 V		26	40	mΩ
Low-side Switch Resistance(1)	VIN = 12 V		19	30	mΩ
ERROR AMPLIFIER					
Error amplifier Transconductance (gm)	–2 µA < I_{COMP} < 2 µA, $V_{(COMP)}$ = 1 V		1300		µMhos
Error amplifier DC gain	VSENSE = 0.8 V	1000	3100		V/V
Error amplifier source/sink	$V_{(COMP)}$ = 1 V, 100-mV input overdrive		±110		µA
Start switching threshold			0.25		V
COMP to Iswitch gm			16		A/V
CURRENT LIMIT					
High-side switch current limit threshold		8	11		A
Low-side switch sourcing current limit		7	10		A
Low-side switch sinking current limit			2.3		A

Figure 1: Sample output from an integrated circuit [1]

II. Literature Review

This project was a continuation of Project 1 from the course DS6372 "Applied Statistics", Spring 2019 at Southern Methodist University [2]. The linear regression models obtained from project 1 suffered from assumption violations, the most severe being the equal variance assumption. The normality of the residuals was also a concern since the residuals were right skewed, although this violation was not severe due to the large sample size. It was also observed that 2-way interaction of all features was not practical without dimensionality reduction since it would create roughly 28,680 predictors with only 6980 observations. Another interesting observation was that a few observations in the training set were high leverage points and this may have impacted the accuracy of the final model.

From project 1, we identified a few possible improvements that could be made to the model fit. These improvements pointed to the need for either (1) performing intelligent/selective feature engineering and variable selection using domain expertise, or (2) using non-parametric models such as tree-based models.

Both the previous and this project had the constraint that only linear/logistic regression models along with clustering algorithms and dimensionality reduction techniques could be used. Hence, in this project, we only acted upon the first idea outlined above.

III. Data Description and Collection

Data for this project was sponsored by Texas Instruments Inc. (TI). Due to proprietary nature of the information, the variables were anonymized. The true identity of the variables was known to only one of the authors of this paper through their association with TI. This information was important as it was used to do the selective feature engineering that will be discussed later in this paper.

The data consisted of 10,000 observations capturing the performance of an integrated circuit under various conditions. There were 240 features consisting of:

(1) Engineer-controlled variables (x1 – x23): Values for these variables were spread across a large range; some were in the range of 1 to 100 while others were in the Nano or Micro range (Figure 2).
(2) Process variation variables (stat1 – stat217): These parameters are beyond human control. They represent various statistical manufacturing parameters. The variables varied between -3 and 3 with a mean of 0 representing the ±3 sigma variation around the mean (typical) process (Figure 2).
(3) Output Variables (y1 - y19) which represented various output variables.

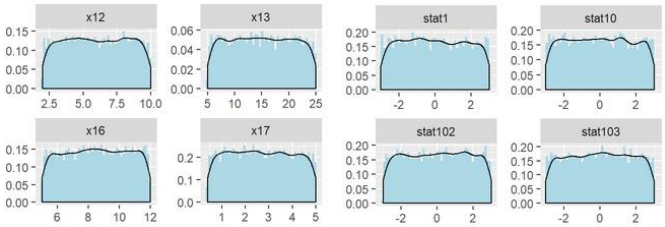

Figure 2: Sample distribution of few engineer-controlled and statistical predictors

The engineer-controlled variables have a predefined range of values that the engineer can chose from. Since they can pick any value in this range, the values for these variables were uniformly and randomly sampled from the range of acceptable values while the data was being collected. Statistical features were also uniformly randomly sampled since the goal was to obtain good model accuracy throughout the statistical variation range and not just closer to the population means (which would have been the case if the data was sampled using a gaussian distribution since in that case, the training data would have had more points closer to the mean and very few points at the ± 3 sigma level).

IV. Research Methodology

A. Output Selection

After discussion with subject matter experts from Texas Instruments, it was decided to focus on modeling 'y3' as it was a critical output of this integrated circuit.

B. Modeling Approach

From project 1, we realized that the use of the predictors "as-is" may not lead to the desired prediction accuracy. Hence, we tried using 2-way interactions of the predictor variables in various forms to improve the model fit. This included trying (1) 2-way interactions of only the engineer-controlled variables with statistical variables used as-is without interactions followed by Principal Component Analysis (PCA), and (2) 2-way interactions of all 240 predictors followed by PCA. As we will see shortly, these did not lead to an improvement in the results.

Finally, we performed intelligent feature engineering using semiconductor domain expertise. This included looking at theoretical equations from semiconductor theory and creating the necessary variables from the ones that were available. Once these variables were created, we performed 2-way interaction of all the variables. As discussed, earlier, this leads to more predictors than observations. Hence, we followed this with dimensionality reduction using PCA. The reduced Principal Components (PCs) were then used to build a linear regression model. First a full model was created using the filtered PCs and this was followed by variable selection using LASSO and LARS. Forward, Backward and Stepwise selection were not used in this project because these algorithms were taking a long time to run due to the large number of predictors even after dimensionality reduction.

R programming language was used to perform all the analysis. We used the caret package [3] to perform and manage the training and cross validation process. Specifically, we used the leaps [4], glmnet [5] and lars [6] libraries with the caret framework in order to perform variable selection. In an effort to make this research reproducible, we have provided the entire code on GitHub [7]

C. Good Modeling Practices

When building the predictive models, we wanted to avoid overfitting the training dataset. This was especially critical since we were expanding our predictors using 2-way interactions. Even though we eventually followed this with PCA which reduced the number of predictors, we were still at risk of overfitting with such a large number of predictors. To avoid overfitting, we used an 80:20 ratio split on our dataset to develop the Train and Test sets and verified the lack of overfitting using the test RMSE. In addition, during the training process, a 10-fold cross validation technique was used for model training and selection.

V. Exploratory Data Analysis

A. Data Preparation

Like most real-world datasets, this one also needed some cleaning. Basic descriptive statistics revealed that 3020 NA values were present for y3. After consulting with the expert from TI, we found that the predictors (features) for these data points were not practical in combination with each other. Hence, these points are not valid and could be removed without impacting the predicting power of the model being developed.

Note that if these were indeed valid data points, we could not have simply removed them from the dataset since it would have violated the random sampling we performed initially, and this would have affected the generalization of the model to the entire design space (population).

B. *Output*

We began exploratory data analysis on the target variable y3 and noticed right skewness in the distribution. We performed a log transformation on this variable which made the data less skewed. Hence, we proceeded with the log transformed variable "y3.log" for our analysis (Figure 3).

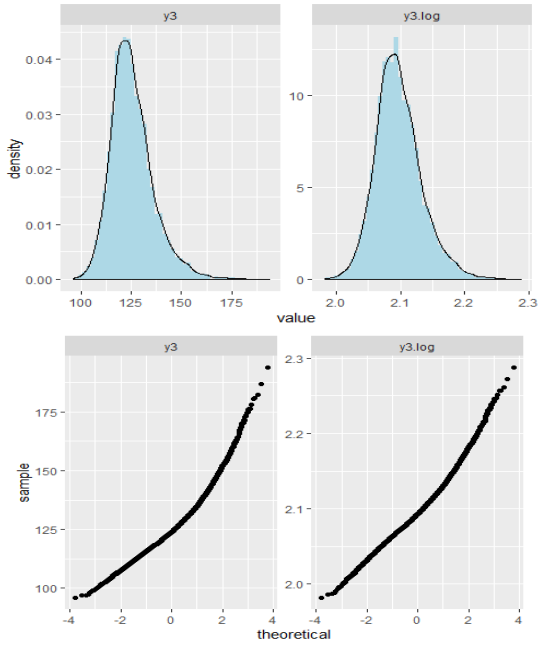

Figure 3: Histogram and QQ plot of y3 and log(y3)

C. *Input Predictors*

To determine if there is a correlation within predictors (features), we checked for multicollinearity. Since inputs were randomly selected independently, we did not expect there to be multicollinearity. After running the analysis, the VIF values (Figure 4) confirmed that there was no issue with multicollinearity as all VIF values were < 10.

##	Variables	VIF	##	Variables	VIF
## 1	stat202	1.063592	## 6	stat184	1.059400
## 2	stat141	1.062435	## 7	stat70	1.058888
## 3	stat52	1.062123	## 8	stat150	1.058825
## 4	stat178	1.062030	## 9	stat14	1.058728
## 5	stat164	1.059900	## 10	stat37	1.058385

Figure 4: Top 10 predictors by VIF

D. *Transformations*

To improve the prediction capability of the model, intelligent feature engineering was performed with intuition derived from basic semiconductor theory. New features such as x2/x1, x6/x5, log(x23), log(x11), $(1/x1)^2$, etc. were created and used subsequently in the models. In all, 41 such features were created using the existing variables (Figure 5).

```
data$x2byx1 = data$x2/data$x1
data$x6byx5 = data$x6/data$x5
data$x9byx7 = data$x9/data$x7
data$x10byx8 = data$x10/data$x8
data$x14byx12=data$x14/data$x12
data$x15byx13=data$x15/data$x13
data$x17byx16=data$x17/data$x16
data$x19byx18=data$x19/data$x18
data$x21byx20=data$x21/data$x20
data$x23byx22=data$x23/data$x22
data$x1sqinv = 1/(data$x1)^2
data$x5sqinv = 1/(data$x5)^2
data$x7sqinv = 1/(data$x7)^2
data$x8sqinv = 1/(data$x8)^2
data$x12sqinv = 1/(data$x12)^2
data$x13sqinv = 1/(data$x13)^2
data$x16sqinv = 1/(data$x16)^2
data$x18sqinv = 1/(data$x18)^2
data$x20sqinv = 1/(data$x20)^2
data$x22sqinv = 1/(data$x22)^2
data$x1log = log(data$x1)
data$x2log = log(data$x2)
data$x5log = log(data$x5)
data$x6log = log(data$x6)
data$x7log = log(data$x7)
data$x9log = log(data$x9)
data$x8log = log(data$x8)
data$x10log = log(data$x10)
data$x12log = log(data$x12)
data$x14log = log(data$x14)
data$x13log = log(data$x13)
data$x15log = log(data$x15)
data$x16log = log(data$x16)
data$x17log = log(data$x17)
data$x18log = log(data$x18)
data$x19log = log(data$x19)
data$x20log = log(data$x20)
data$x21log = log(data$x21)
data$x22log = log(data$x22)
data$x23log = log(data$x23)
data$x11log = log(data$x11)
```

Figure 5: Domain specific feature engineered predictors

VI. Model Results

As outlined in the research methodology section, we first tried taking 2-way interactions of only the engineer-controlled variables amongst themselves along with the standalone statistical variables (ID 1). In order to not increase the number of predictors significantly, we followed this with Principal Component Analysis (PCA) and Regression (PCR). However, this did not lead to much improvement in the model fit (Figure 7 compared with Figure 6; green and red lines represent ±10% and ± 15% difference from actual values respectively).

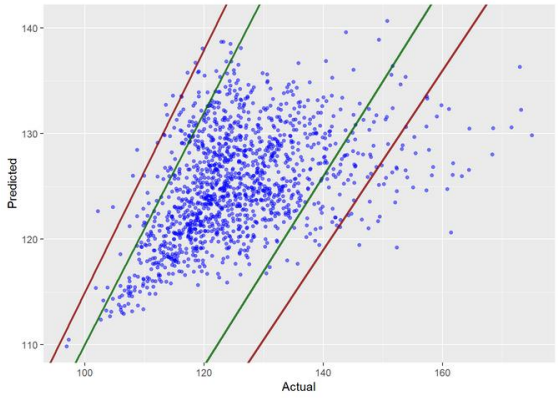

Figure 6: Predicted vs. Actual Values (Project 1 best model)

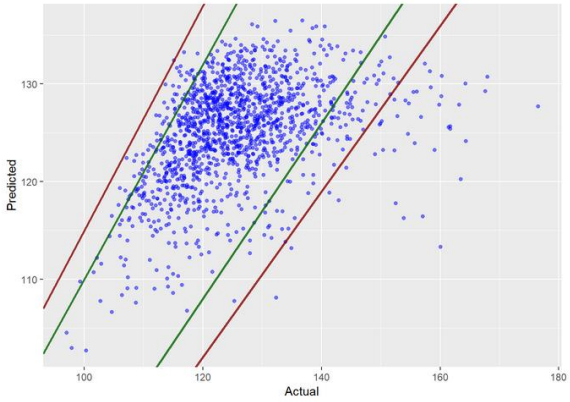

Figure 7: Predicted vs. Actual Values (Full Model 2-way with engineer-controlled + standalone statistical variables)

Next, we tried 2-way interaction of all the predictors followed by PCA and PCR (ID 2). This model offered some improvements in the results and did not suffer from the same assumption violations as the model from project 1. The

residuals were normally distributed (Figure 8) and showed equal variance across all predicted values (Figures 9). The train R2 was also better at 0.786 (Table 1). However, the test RMSE saw a degradation in performance to 16.76 and the same is visible in Figure 10. Using LASSO (ID 3) and LARS (ID4) for regularization, we were able to avoid overfitting to some extent and get the test RMSE down to 9.91 (Table 1).

This led us to 2 conclusions – (1) In order to get better train fit statistics, we needed to consider 2-way interaction of all variables, not just the engineer-controlled variables, and (2) Variable selection and regularizations techniques were necessary to prevent overfitting.

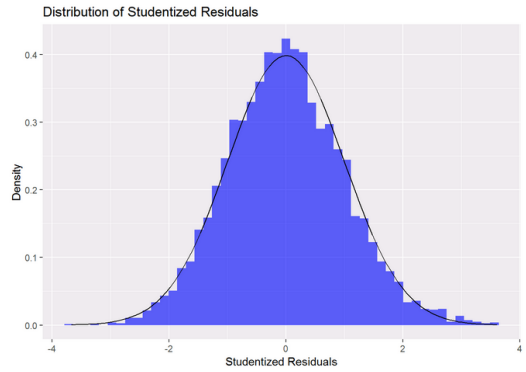

Figure 8: Histogram of Residuals

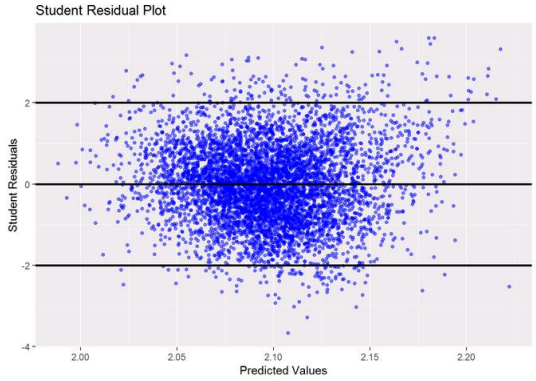

Figure 9: Studentized Residuals vs. Predicted Values

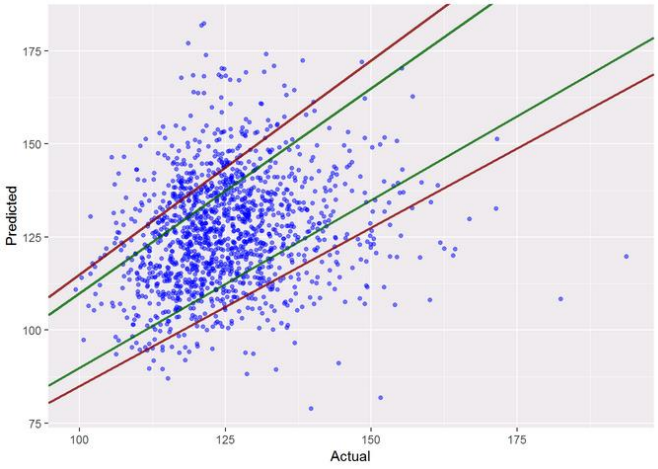

Figure 10: Predicted vs. Actual Values (Full Model 2-way interaction of all variables)

Finally, we used the intelligent feature engineered variables along with the original variables as our new set of predictors. While the original data set did not show any multicollinearity, the new dataset showed considerable multicollinearity since many features were derived from the others. This is clearly visible in Figure 11. In addition, since the model in project 1 did not give a good fit as is, we performed a 2-way interaction of all the features with each other. This increased the number of predictors from 240 to 39,340. We solved the issue of multicollinearity as well as the extremely high dimensionality by using Principal Component Analysis. PCA transformed the correlated inputs into a set of non-correlated linear combinations which resolved the multicollinearity issue. In addition, the first 3455 principal components accounted for 80% of the variability in the predictors (Figure 12). This was less than 9% of the dimensions obtained after the 2-way interaction. Hence, we decided to forgo the remaining PCs and subsequently used only the first 3455 PCs in our Linear Regression modeling.

##	Variables	VIF	##	Variables	VIF
## 1	x16log	3898.657885	## 26	x22	89.101209
## 2	x5log	2548.241629	## 27	x7sqinv	67.107315
## 3	x20log	1933.938073	## 28	x1sqinv	65.708942
## 4	x16	1794.335885	## 29	x21	57.729060
## 5	x11	1617.835412	## 30	x2	48.327951
## 6	x11log	1617.529796	## 31	x13sqinv	46.881321
## 7	x8log	1616.618846	## 32	x21log	46.673228
## 8	x5	1181.352484	## 33	x12sqinv	45.232723
## 9	x20	899.832193	## 34	x2log	43.040667
## 10	x8	760.784111	## 35	x18sqinv	40.950701
## 11	x7log	551.537736	## 36	x23	38.225244
## 12	x1log	539.431432	## 37	x23log	35.673242
## 13	x16sqinv	449.246006	## 38	x6	34.791262
## 14	x13log	379.929844	## 39	x17	28.762744
## 15	x12log	371.951963	## 40	x21byx20	23.217635
## 16	x18log	316.819341	## 41	x22sqinv	23.209779
## 17	x5sqinv	296.949745	## 42	x6log	22.007646
## 18	x7	267.551246	## 43	x17byx16	21.586372
## 19	x1	259.322495	## 44	x6byx5	20.627254
## 20	x20sqinv	226.688625	## 45	x9	20.143701
## 21	x8sqinv	190.358246	## 46	x19	19.326069
## 22	x13	186.168906	## 47	x10	19.180606
## 23	x12	185.412904	## 48	x14	17.547345
## 24	x22log	176.081153	## 49	x2byx1	17.031845
## 25	x18	158.571673	## 50	x10byx8	15.083700

Figure 11: Top 50 variables by VIF show multicollinearity

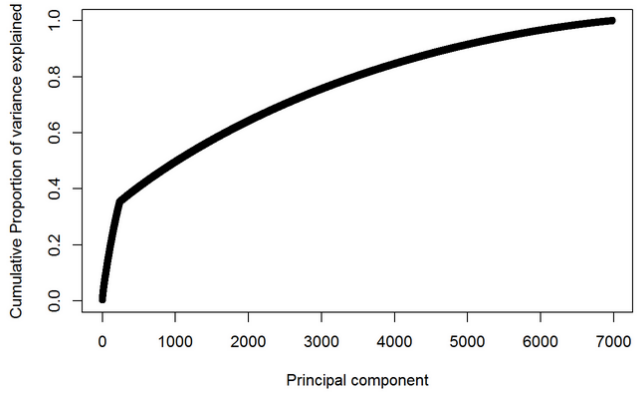

Figure 12: Scree plot showing cumulative variance explained by the first 7000 principal components

Next, we build a full model with these 3455 PCs and log(y3) as the output (ID 5). As with model ID 2, this model's residual analysis also showed that it did not exhibit the same shortcoming as the model from project 1. The residuals were normally distributed as shown by the histogram (Figure 13). The studentized residuals also show almost constant variance, unlike that in project 1, with very few residuals outside the ±2 sigma lines (Figure 14) which is consistent with a normal distribution (95% points between the ±2 sigma lines). The model exhibited very good fit statistics with an adjusted R^2 of 0.938 (R^2 = 0.976).

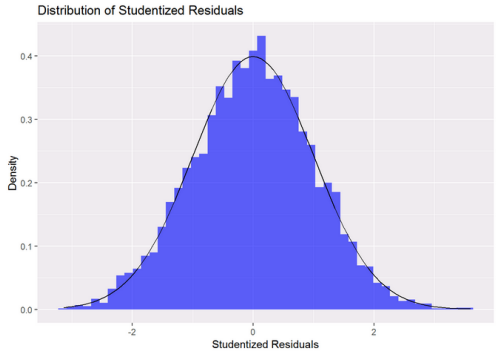

Figure 13: Histogram of Residuals

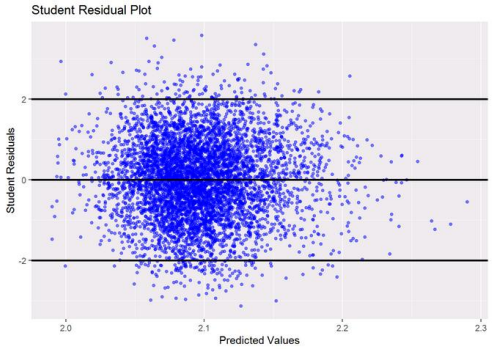

Figure 14: Studentized Residuals vs. Predicted Values

The initial fear of overfitting with so many predictors was laid to rest by looking at the test set predictions. The test RMSE obtained was just 4.59 (Table 1) which was well within the target range (<10%). The scatter plot of the predicted vs. actual values (Figure 15) for the test set also shows that very few points exceed the ±10% limit (green lines) and none of the points exceed the ±15% limit (red lines). Hence, this full model satisfied all the prediction requirements. This was not the case in the original model from project 1 (Figure 6).

We went further and performed variable selection and regularization on this full model to remove any scope of overfitting using LASSO (ID 6) and LARS (ID 7). Both these models provided similar performance metrics on the tests set and both offered some improvement in the test RMSE compared to the full model as shown in Table 1. The Train R^2 was lower for both these models compared to the full model, but since the goal was prediction, the test RMSE was a more appropriate metric to consider. Given this metric, we choose the LASSO model as our final model. We can see from the scatter plot of predicted vs. actual values (Figure 16) for the LASSO model that none of the predicted points not lie outside the ±10% limits and hence we conclude that this model offers better prediction than the full model.

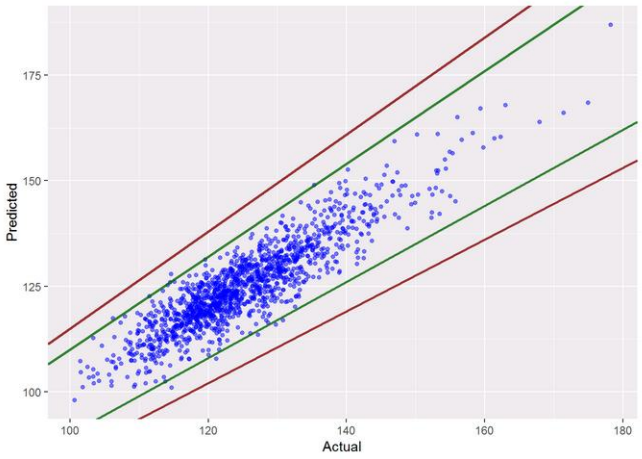

Figure 15: Predicted vs. Actual Values (Full Model)

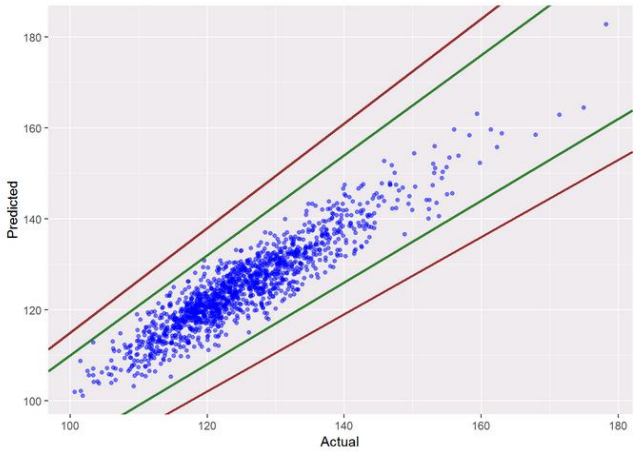

Figure 16: Predicted vs. Actual Values (LASSO)

TABLE I
COMPARISON OF DIFFERENT MODELS

ID #	MODEL (WITH PCA)	# VARS	TRAIN R^2	TEST RMSE
1	*FULL MODEL (2-WAY EC + STAT)*	164	0.266	9.45
2	*FULL MODEL (2-WAY ALL)*	3834	0.786	16.76
3	*LASSO (2-WAY ALL)*	3834	0.139	9.91
4	*LARS (2-WAY ALL)*	3834	0.139	9.91
5	*FULL MODEL (FEAT ENG. 2-WAY ALL)*	3455	0.976	4.59
6	*LASSO (FEAT ENG. 2-WAY ALL)*	3455	0.864	3.84
7	*LARS (FEAT ENG. 2-WAY ALL)*	3455	0.864	3.85

VII. DISCUSSION

While we were able to achieve the prediction goal set out for this project using this approach, we recognize that the said method has some issues in terms of scalability. Due to the large number of dimensions involved, performing PCA and variable selection is extremely computation intensive. In fact, the model development failed to run on a laptop with 4 multithreaded cores with 16GB of RAM. The model had to eventually be run on a more powerful machine (32 cores and

256 GB of memory) and even then, it took over 6 hours of compute time (for PCA, LASSO and LARS combined) with RAM usage peaking at over 60GB. Stepwise variable selection was not performed since it ran for more than 7 days on this machine (without completing). This would be severely limiting if the model development must be scaled to hundreds of outputs and integrated circuits.

One suggestion to alleviate the scalability issue to multiple outputs is to use a cloud computing platform such as Amazon Web Service where multiple instances can be launched simultaneously, but that was beyond the scope of this project.

VIII. Conclusion

Through this project, we have shown that even though semiconductor physics offers a highly non-linear design space, we can still use linear regression techniques to obtain a reasonable model fit. The key to this process is incorporating intelligent domain specific feature engineering, including higher order interaction terms and possibly employing dimensionality reduction techniques if including higher ordered terms results in too many predictors.

This method is not without its challenges though, the most severe being the computation requirements. In light of these findings, other non-parametric tree-based models (such as Random Forest, XGBoost) or Artificial Neural Networks could also be trained on this dataset to see if we can obtain a similar or better fit within a reasonable timeframe. That though is left as a follow-up to be performed at a later time.

IX. References

[1] "Texas Intruments Inc.," [Online]. Available: http://www.ti.com/lit/ds/symlink/tps54620.pdf.

[2] N. Gupta, M. Moro and J. Duran, "Predicting Performance of Integrated Circuits using Regression Analysis," 2019.

[3] M. Kuhn, "Building Predictive Models in R Using the caret Package," *Journal of Statistical Software*, vol. 28, no. 5, pp. 1-26, 2008.

[4] T. Lumley, *leaps: Regression Subset Selection*, 2017.

[5] J. Friedman, T. Hastie and R. Tibshirani, "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1-22, 2010.

[6] T. Hastie and B. Efron, "lars: Least Angle Regression, Lasso and Forward Stagewise," 2013.

[7] "GitHub Repository," [Online]. Available: https://github.com/ngupta23/MSDS-6372-Project2/tree/master/Analysis/Master-Submitted.