[[1]](#footnote-2)

Predicting Performance of Integrated Circuits using Machine Learning

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

**Abstract — Semiconductor manufacturing is a variable process and outcomes depend on several factors. To meet target specifications, some parameters are controlled 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, Quadratic Discriminate Analysis (QDA), Logistic Regression.*

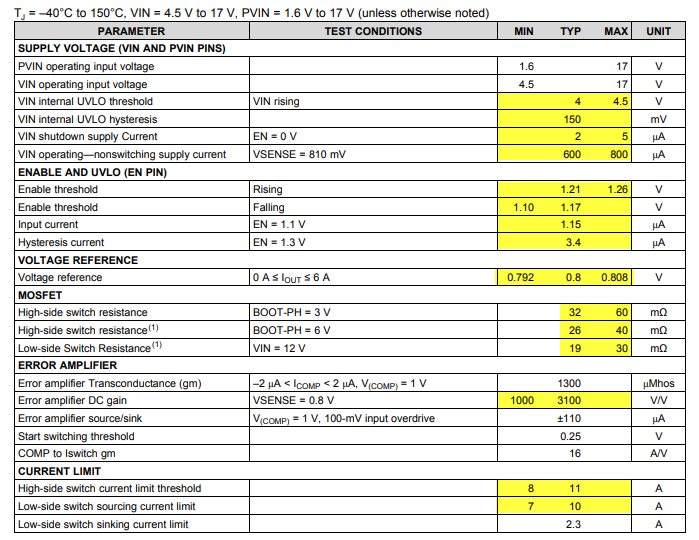
# INTRODUCTION

L

ike in most 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 semiconductor manufacturing, the answer is yes. Current practice is to use electrical simulation (plus running Monte Carlo simulations) to identify specification 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 can 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.



*Fig 1: Sample output from an integrated circuit* [*Reference*](http://www.ti.com/lit/ds/symlink/tps54620.pdf)

# Literature Review

This project is a continuation of Project 1 for course DS6372 “Applied Statistics”, Spring 2019 [**XX**]. 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) building a model to predict high influence points from the available training data, predicting if a new observation belongs to this group and finally, using this categorical prediction as an additional input to the eventual linear regression model, or (3) 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 acted upon the first idea and touched upon the second one to some extent.

# 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.
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 representing the ±3 sigma variation around the mean (typical) process.
3. Output Variables (y1 - y19) which represented various output variables.

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).

# Research Methodology

## 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.

## Modeling Approach 1

For this project, we tried 2 different approaches. The first one involved performing 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 Principal Component Analysis (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.

## Modeling Approach 2

In the second approach, we build the initial steps of a model pipeline. We used the full model from project 1 to label the training observations as high leverage or low leverage. This data was then used to train various classification models including Linear/Quadratic Discriminate Analysis (LDA/QDA) and Logistic Regression models. The first step of the proposed pipeline involved classifying new observations into 2 levels, high leverage or low leverage, using the classification models built. Once classified, the predicted class level could then be used as a predictor for the second and final stage of the pipeline which would be the linear regression model for predicting y3. The hope was that by introducing this categorical predictor in the linear regression model, we would intrinsically build 2 separate regression lines – one each for the high and low leverage points and that this would lead to better prediction accuracy.

## 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 of all variables. Even though we eventually follow this with PCA which reduces the number of predictors to XX, we are still at risk of overfitting with such huge number of predictors. To avoid overfitting, we used an 80:20 ratio split on our dataset to develop the Train and Test sets. In addition, during the training process, a 10-fold cross validation technique was used for model selection.

# Exploratory Data Analysis

## 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 (Fig 2). 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.

message('Original cases: ',nrow(data.ori))

## Original cases: 10000

message('Non-Complete:', nrow(data.notComplete))

## Non-Complete: 3020

*Fig 2: Original data and observations with missing values*

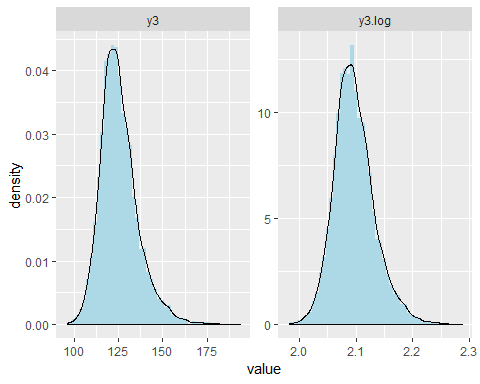
Note that if these were indeed valid data points, we could not have simply removed them from the dataset since it would have violated the random sampling we performed initially, and this would have affected the generalization of the model to the entire design space (population).

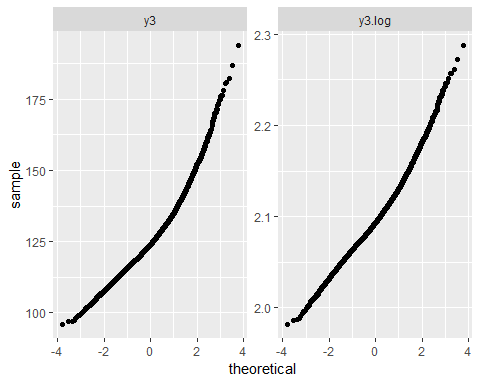
## Output

We began exploratory data analysis on the target variable y3 and noticed right skewness, therefore we performed a log transformation. Log transformation made the data a little more skewed, therefore we proceeded with the log transformed variable “y3.log” (Fig 3).

## Input Predictors

VIF and Correlations: To determine if there is a correlation within predictors (features), we began by checking for multicollinearity. Since inputs were randomly selected, we did not expect there to be multicollinearity. After running the analysis, the VIF values (Fig 4) confirmed that there was no issue with multicollinearity.



**

*Fig 3: Histogram and QQ plot of y3 and log(y3)*

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

*Fig 4: Top 10 predictors by VIF*

## Transformations

To improve the prediction capability of the model, intelligent feature engineering was performed in approach 1 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 model. In all, 41 such features were created with the existing variables (Fig XX).

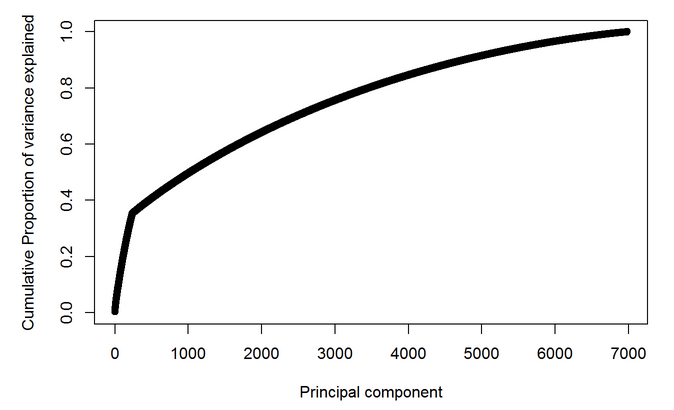
|  |  |
| --- | --- |
| data$x2byx1 = data$x2/data$x1  data$x6byx5 = data$x6/data$x5  data$x9byx7 = data$x9/data$x7  data$x10byx8 = data$x10/data$x8  data$x14byx12=data$x14/data$x12  data$x15byx13=data$x15/data$x13  data$x17byx16=data$x17/data$x16  data$x19byx18=data$x19/data$x18  data$x21byx20=data$x21/data$x20  data$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) |

*Fig XX: Domain specific feature engineered predictors*

# Model Results

## Modeling Approach 1

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 Fig XX. 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). PCA transformed the correlated inputs into a set of non-correlated linear combinations which resolved the multicollinearity issue. In addition, the first few principal components account for most of the variability in the predictors. If we forgo a small amount of variability in the predictors, we can reduce the dimensionality drastically. As can be seen from Fig XX, the first 3,455 principal components account for ~ 80% of the data variability and this is less than 9% of the dimensions obtained from the 2-way interaction. Subsequently, we used these 3455 PCs in our Linear Regression modeling.

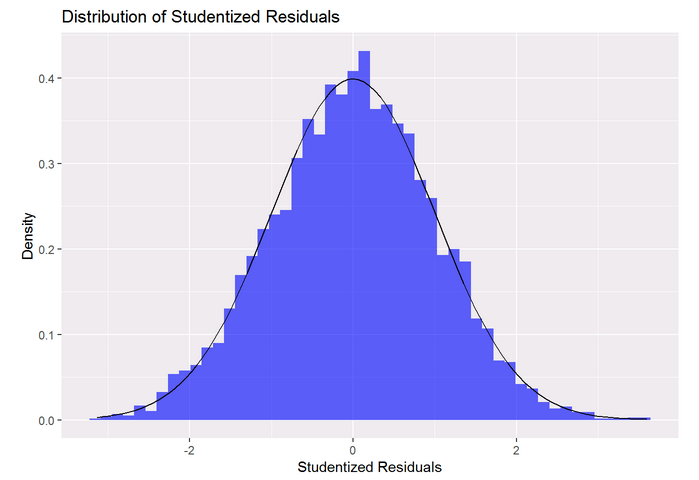


*Fig XX: Scree plot showing cumulative variance explained by the first 7000 principal components*

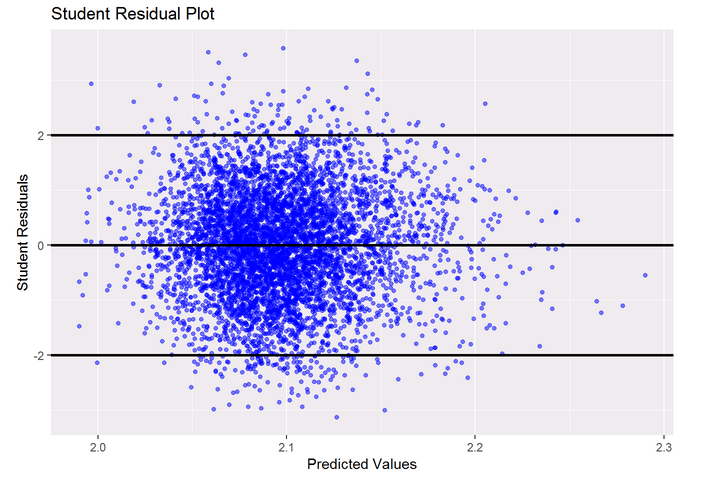
|  |  |
| --- | --- |
| **## Variables VIF**  ## 1 x16log 3898.657885  ## 2 x5log 2548.241629  ## 3 x20log 1933.938073  ## 4 x16 1794.335885  ## 5 x11 1617.835412  ## 6 x11log 1617.529796  ## 7 x8log 1616.618846  ## 8 x5 1181.352484  ## 9 x20 899.832193  ## 10 x8 760.784111  ## 11 x7log 551.537736  ##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

*Fig XX: Top 50 variables by VIF show multicollinearity*

Next, we build a full model with these 3455 PCs and log(y3) as the output. The model residual analysis showed that this full model did not exhibit the same shortcoming as the model from project 1. The residuals were normally distributed as shown by the histogram (Fig XX). The studentized residuals also show almost constant variance unlike that in project 1 with very few residuals outside the ±2 sigma lines (Fig XX) which is consistent with a normal distribution (95% points between the ±2 sigma lines). The model exhibited very good fit statistics with an adjusted R2 of 0.938.

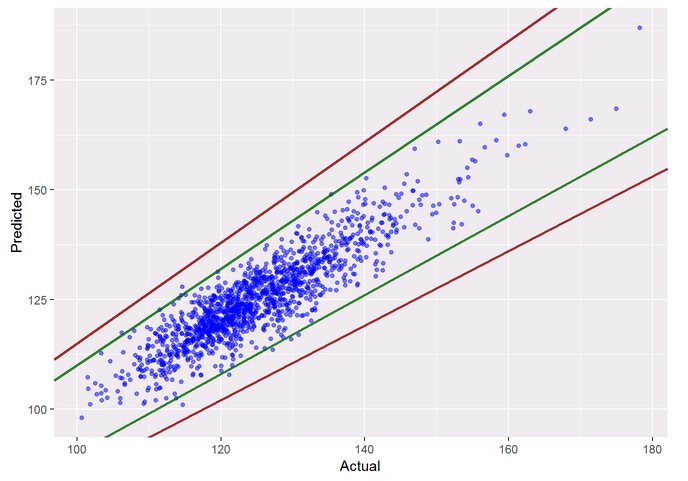


*Fig XX: Histogram of Residuals*

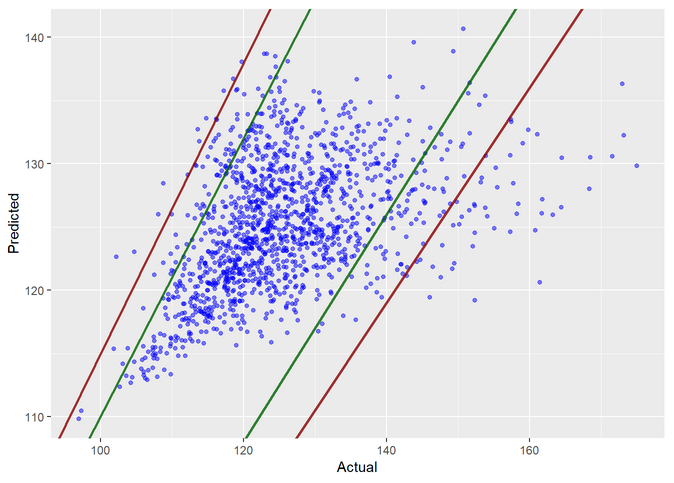


*Fig XX: Studentized Residuals vs. Predicted Values*

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 (Fig XX) which was well within the target range (<10%). The scatter plot for the predicted vs. actual values (Fig XX) 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 original prediction requirements. This was not the case in the original model from project 1 (Fig XX).



*Fig XX: Predicted vs. Actual Values (Full Model)*



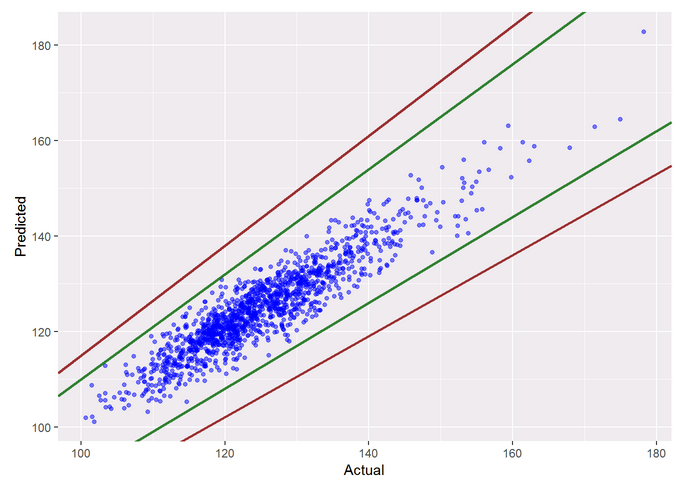
*Fig XX: Predicted vs. Actual Values (Project 1 Best Model)*

We went further and performed variable selection and regularization on this full model to remove any scope of overfitting. Both LASSO and LARS provided similar performance metrics on the tests set and both offered some improvement in the test RMSE as shown in Table XX. The Train R2 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 (Fig XX) 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.

TABLE I

Comparison of Different Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Model** | **# Variables** | **Train R2** | **Test RMSE** |
|  | *Full Model* | *3455* | *0.976* | *4.59* |
|  | *LASSO* | *3455* | *0.864* | *3.84* |
|  | *LARS* | *3455* | *0.864* | *3.85* |
|  |  |  |  |  |



*Fig XX: Predicted vs. Actual Values (LASSO)*

## Modeling Approach 2

# Discussion

## Modeling Approach 1

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 (add configuration) and even then, it took over XX hours of compute time with RAM usage peaking at XX GB. This would be severely limiting if the model development must be scaled to hundreds of outputs and integrated circuits.

One suggestion to alleviate this issue is to use a cloud computing platform such as Amazon Web Service but that was be scope of this project.

## Modeling Approach 2

While we propose a 2-stage pipeline for this modeling approach, we recognize an underlying issue. The first stage of the pipeline involved training a logistic regression model. However, the outputs of the observations were labelled from the high leverage points from the project 1 full model. This was a subjective choice and the identification of the high leverage points can itself vary and depend on the fit of the original model.

Hence, although a good exercise in theory, this would not be of practical significance when implementing a predictive system at scale.

# 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. That though is left as a follow-up to be performed at a later time.

# References

|  |  |
| --- | --- |
| [1] | "Hadoop MapReduce Tutorial," [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred\_tutorial.html. |
| [2] | Zaharia, Matei, et al., "Fast and interactive analytics over Hadoop data with Spark," Usenix Login, 2012. |
| [3] | Spark, "Spark API Documentation," [Online]. Available: https://spark.apache.org/docs/2.3.1/api.html. |
| [4] | P. Gandhi, "Apache Spark : Python vs. Scala," KDnuggets, May 2018. [Online]. Available: https://www.kdnuggets.com/2018/05/apache-spark-python-scala.html. |
| [5] | Nti-Addae, Yaw; et al., "Benchmarking Database Systems for Genomic Selection Implementation," bioRxiv, 2019. |
| [6] | N. Gureev, "Hive, Spark, Presto for Interactive Queries on Big Data," KTH Royal Institute of Technology, Stockholm, 2018. |
| [7] | S. Akhadov, "PySpark at Bare-Metal Speed," ETH, Zurich, 2017. |
| [8] | Databricks, "Databricks FAQ," Databricks, 2018. [Online]. Available: https://databricks.com/product/faq. [Accessed 15 03 2019]. |
| [9] | DWYL, [Online]. Available: https://github.com/dwyl/english-words. [Accessed 10 03 2019]. |

1. [↑](#footnote-ref-2)