Using Machine Learning to Predict Fabric Production and Rejection Amounts

Hailey Johnson, Nathan Mitchell

Introduction:

Production and Rejection are two of the most important metrics in the textile industry. As such, being able to predict these values would be incredibly useful for business management. We aim to expand on research by Toufique Ahmed and Shihab Uddin (2023) with the goal of predicting the amount of fabric both produced and rejected from a weaving factory. These data were originally collected from Evince Textiles Ltd. in Gazipur, Bangladesh, from January to September of 2013. The researchers Ahmed and Uddin performed merging, preprocessing, filtering, and feature engineering on the original data. We will be using various machine learning methods on their finalized dataset and comparing the methods to find the optimal procedure for predicting production and rejection amounts.

In the original paper, the creators of this dataset noted an interest in someone using statistics, data science, and machine learning in order to predict these two outcome variables, but also that no one had taken the opportunity to do so. With further research, we were able to verify this claim. At the time of writing this report, only two other papers cite this article.

After looking through these two articles, we were able to verify that neither of them had used this dataset at all. The first article that cited Ahmed & Uddin (2023) proposed another dataset to be used. We assumed that this new dataset may have contained the one we were planning on using, but, upon further inspection, this paper cites Ahmed & Uddin (2023) as being "valuable contributors" to the field of textile fiber analysis (Pereira et al., 2024).

The second article mentioned they used an optimized backpropagation neural network, but it appears that they performed their analysis on a different dataset as they were interested in predicting “titanium dioxide nanomaterials concentration for purification industrial wastewater” (Hassanien et al., 2023). Further reading the article, we saw that the only mention of Ahmed & Uddin (2023) was in their reason for pursuing this analysis — citing the fact that AI and data science can be used in this field, though not much work has been done on that subject.

Knowing that no one else has published research using this dataset, we were free to choose any method that we saw fit to answer the questions laid out by Ahmed & Uddin. However, before we discuss methods and results, we will provide some background on the variables that are present within the dataset. After data collection, processing, and feature engineering, they ended up with a dataset that contained 14 different variables that each had 22,010 observations that were taken across 272 days. This dataset had been condensed from a much larger dataset, which contained 18 variables and 121,148 observations, in order to remove many of the missing values and retain only the most important variables for machine learning. In our goal of prediction, we will be focusing on the two primary

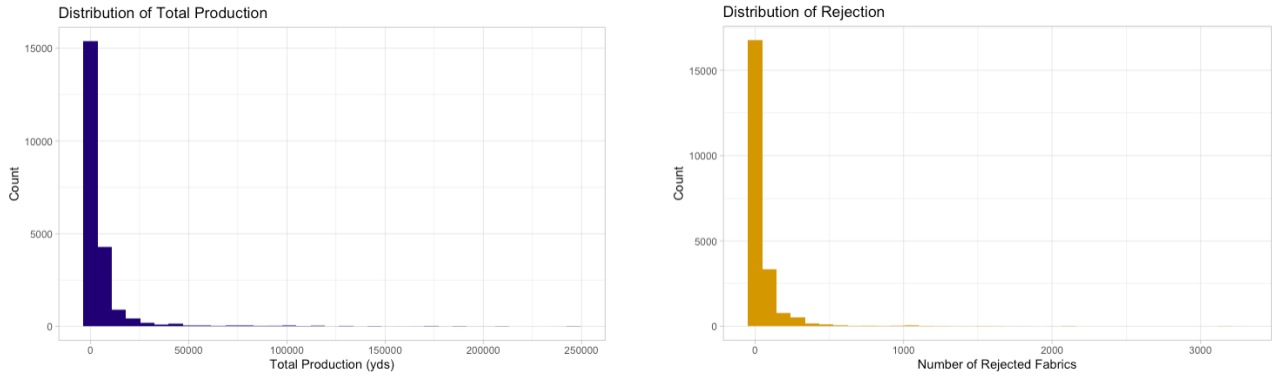

Figure I: Outcome Variable Distributions

response variables: Total Production and Rejection. Total Production refers to the total amount of fabric that was produced that day while Total Rejection refers to the total amount of fabric that was rejected that day due to various fabric faults (Ahmed & Uddin, 2023).

With the two response variables set aside, we were left with 12 variables that may be used for prediction.

Preprocessing:

Ultimately, we decided to keep nine predictor variables. These are Req_Finish_Fabric, which measures the total amount of fabrics required in length, Fabric_Allowance, which measures the fabric allowance, Rec_beam_length(yds), the total length of the yarn supplied, Shrink_allow, which measures the shrinkage allowance, Req_grey_fabric, which measures the calculated amount of gray fabric needed, Req_beam_length(yds), which is a calculated amount of warp yarn needed for the total required fabrics, weft_count, which measures the weft count, EPI, which measures the ends per inch, and PPI, which measures picks per inch (Ahmed & Uddin, 2023).

After we had a final dataset determined, we turned to deciding upon methods to use to predict the outcomes. As a baseline, we wanted to try Linear Regression, but, in choosing Linear Regression, we needed to examine our data to determine if they met the required assumptions. To do this, we first examined the distributions of each of the outcome variables we were interested in predicting for context.

Figure I features plots of the distributions of each of the two outcome variables — Total Production on the left and Total Rejection on the right. As can be seen, both of these variables share a very similar right-skewed distribution. Most of the values featured in both of the outcome variables are near 0. Because of this, we not only expected to violate the assumptions required for Linear Regression, we expected for it to be challenging to be precise with our estimates/achieve high predictive power.

The first assumption that we checked that is made by using Linear Regression is that the residuals are normally distributed, which can be assessed using a Q-Q plot. The next assumption we checked was that the errors have constant variance, which can be checked by plotting the residual values against the predicted values (Chatterjee & Hadi, 2006).

The third assumption we checked was that all observations are independent of one another. This assumption does not require a plot to diagnose and, given

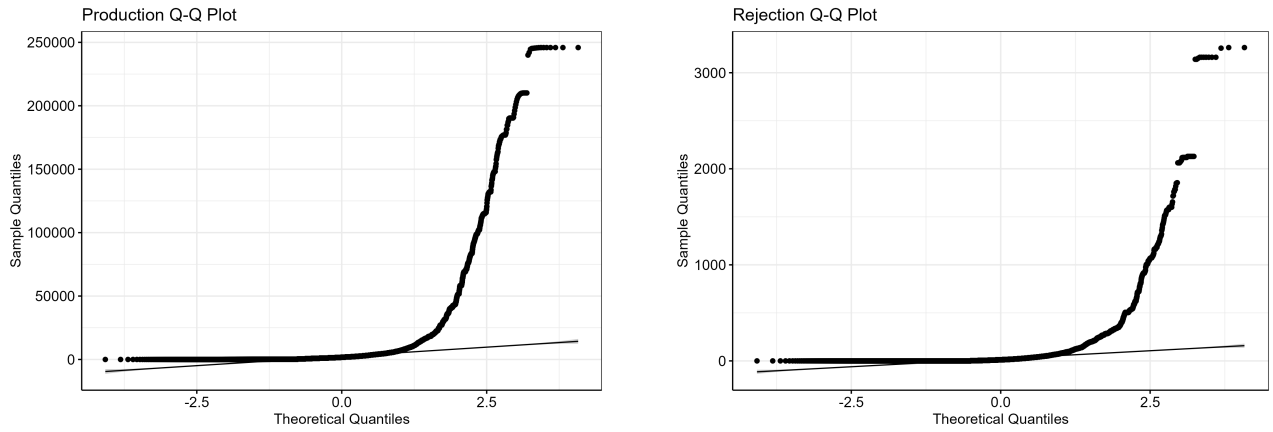

Figure II: Outcome Variable Q-Q Plots

the nature of the data, we will assume that each day's observations are independent. The final assumption that was checked was whether or not it is reasonable to believe that the underlying relationship between the predictors and each outcome variable is truly linear (Chatterjee & Hadi, 2006).

Like the previous assumption, we did not need to create a plot in order to check this assumption. Though we have limited knowledge of textile weaving, it seems accurate to say that our predictors may have a linear relationship with Production. For Rejection, we have no intuition about the underlying relationship.

Now that the assumptions have been discussed, we can show the results of our tests. To assess the normality assumption, we have Q-Q plots for each of the two outcome variables. Shown above in Figure II are two plots — the Q-Q plot for Production on the left and the plot for Rejection on the right.

If there were no violation in the normality assumption, we would expect to see that the actual quantiles align closely with the theoretical quantiles (following the line). As can be seen above for Production, there is a very clear violation. In fact, the violation is so bad that the line the quantiles should be following is nearly flat and barely visible against the plotted quantiles. We didn't expect for Rejection to follow the normality assumption either, but we have it plotted above anyway.

The Q-Q plot for Rejection appears to show just as bad of a violation of the normality assumption as was seen for Production. The actual quantiles do not even remotely follow the line in any capacity. With this, we can confirm that neither of our outcome variables have normally distributed residuals.

Though these results did not bode well for the validity of our remaining assumption, we still wished to check it formally. Thus, below, shown in Figure III, are the plots of residual values against the fitted values for each of our two outcome variables — Total Production on the left and Total Rejection on the right. In order for this assumption to not be violated, we would need to see that the plotted values are roughly uniformly distributed. Evidence of a violation will appear as a funnel shape or nonlinearity around 0.

Given the shape of these two plots, we clearly had another violation. We wanted to see uniformity, but, instead, we can clearly see a funnel shape — a clear indication of a violation of constant variance — in plots for each of the two outcome variables.

Despite three out of the four assumptions being violated, we elected to continue both with Linear Regression and linear models in general (though we will

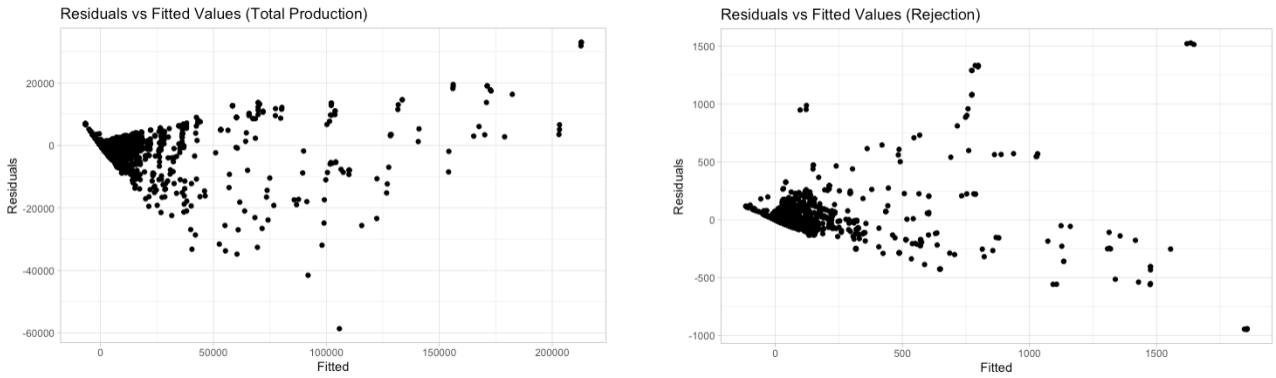

Figure III: Outcome Variable Residual vs. Fitted Values

eventually be relaxing this linearity aspect). In light of this, we want to note that, because of these violations, we will not be drawing any inferential conclusions based on any of these models and, in the case of Linear Regression, will not be disclosing the values of the slope coefficients or intercept value as they should not be interpreted due to the violations. Thus, the remainder of our analysis will be focused solely on prediction and minimizing the predictive error of our two outcome variables.

Our next step was to further process the data into a more usable format. The first step of this was to normalize our predictor variables. We chose not to normalize our outcome variables to keep some interpretability but, given we are not interpreting our predictors (except for one case which will be mentioned) and they are in different units, we chose to standardize them.

In order to get a good sense of whether or not our models were overfitting the data, we elected to split the data into two sets: one for training the model and one for testing the model. By doing this, we could see if a successful model was able to generalize and perform well on unseen data. To do this, we took a random sample of 75% of the data for our training set and the remaining 25% for our testing set. For the remainder of this paper, any estimate of error given will be error calculated by comparing the true values from our testing set and the predicted values of our testing set, i.e. we will be giving testing error.

After we had created our training and testing datasets, we needed to further split the data so that we could fit two different models — one for predicting Total Production and one for Total Rejection. To do this, we split our X training and testing sets, which contained all the variables, into one set that removed Production and another that removed Rejection. For our y sets, we isolated the Production and Rejection columns in each of the training and testing datasets.

Methods:

Despite the fact that our data very clearly violated most of the assumptions made by Linear Regression, we still decided to fit a Multiple Linear Regression (MLR) model for both Total Production and Total Rejection for comparison with our nonlinear methods. Without looking at the coefficients or intercept values, Linear Regression is a method that aims to minimize the sum of the squared distances between the linear line that is fit and the observed y values. The simplicity of this method results in a good baseline for comparison. After fitting the models, what we found was that, when we tried to predict Production, we got an R^2 value of 0.963 — a surprisingly high value — and an R^2 of 0.675 for the model used to predict Rejection.

R^2 is an estimate for how much variability in the

outcome variable can be explained by the predictor variables, but, because of how it is calculated, it can easily be manipulated by data that are not normally distributed or roughly linear. Because of this, and because R^2 isn't a metric available for the other methods that we wish to use, we must discuss another metric to use in order to get a sense of how "good" each model is.

To do this, because we are working on a regression task, we turned to Root Mean Squared Error (RMSE), which can be defined as:

$$\sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}.$$

where n is the number of observations in the dataset, $\hat{y}$ is the predicted value, and y is the observed value. Simply put, RMSE gives us a measure of how far away the model's predictions are from the true outputs. This value, on its own, means very little if only a single model is fit. The real power of using RMSE comes when a) fitting multiple models and b) comparing it to the standard deviation of the outcome variable it is predicting. If one model has one RMSE and the next has a smaller RMSE, it can be said that the second model does a better job at predicting the outcome variable. Similarly, if one model produces an RMSE that is smaller than the standard deviation of that outcome variable and one gives an RMSE that is greater, the model that can explain under the standard deviation has performed better.

Comparing RMSE to the standard deviation of the outcome variable of interest is also helpful for comparing RMSE values for models that are fit to predict variables that have different standard deviations, as dividing by the standard deviation is a way to standardize the estimated error. This means that, even though our outcome variables are in different units, we can easily compare the standardized RMSE values to determine which model does better. We are not aiming to pit models predicting each outcome variable against each other, but being able to compare them is still useful.

Table 1 shows the standard deviation of the testing set of each outcome variable. We will be dividing each RMSE estimate by their respective standard deviation, though we will display both estimates.

Production Std. Dev.	Rejection Std. Dev.
18933.823	155.654

Table 1

With this in mind, we can compare the RMSE values we got for our Linear Regression models to one another. Below, in Table 2, are the RMSE values and their standardized counterparts.

Method	Production RMSE	Rejection RMSE
Linear Regression	3656.91 | 0.187	91.43 | 0.571

Table 2

For the remainder of our discussion, we will display our results in the same way as in Table II — the RMSE on the left and the standardized RMSE on the right for each of the two models.

Given the relative sizes of the R^2 values each model achieved, we were not that surprised by the RMSE that each Linear Regression model was able to achieve. Similar to how the Production model had a larger R^2, meaning the predictors explained more variability in Production than Rejection, the Production model had a much smaller standardized RMSE than the Rejection model. This means that the error garnered by the linear model was much smaller than the standard deviation of the data, meaning the model performed well. Even though the standardized RMSE was smaller for the Production model, the Rejection model still had

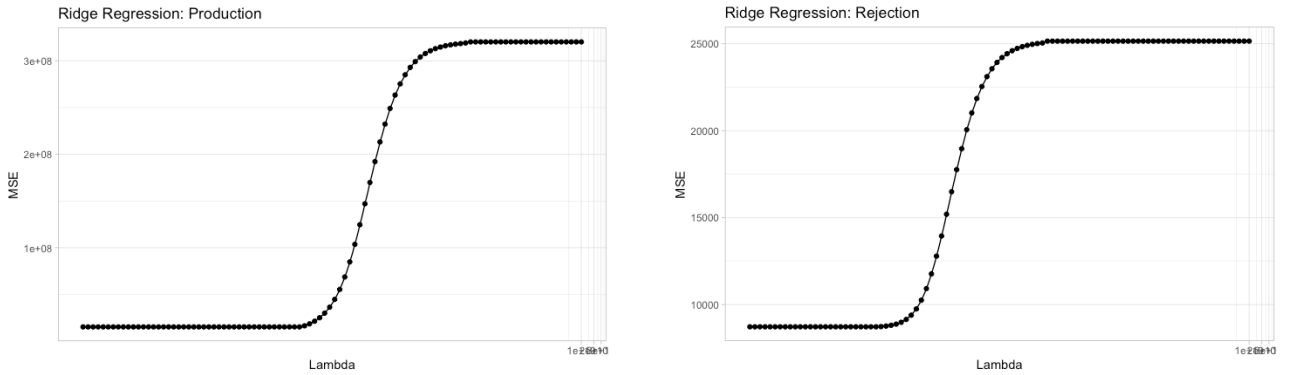

Figure IV: Outcome Variable Distributions

an RMSE that was smaller than its standard deviation. Thus, we can say that each model still was able to perform relatively well.

Moving on from Multiple Linear Regression, we considered both Ridge Regression and LASSO. These techniques, while still linear in nature, are better suited for the goal of prediction specifically than Linear Regression. While typical Linear Regression uses the observed data to obtain maximum likelihood estimates of the regression coefficients, the estimates can be incorrectly inflated when two outcomes have multicollinearity (Murel and Kavlakoglu, 2023). Ridge Regression and LASSO regression work well in these scenarios because they regularize and "shrink" the regression coefficients. This helps to reduce overfitting on the training data, leading to models that are more flexible and perform better on the unseen testing set.

Predictor Variable	VIF
Req_Finish_Fabrics	11618.293
Fabric_Allowance	1.575
Rec_Beam_Length	3.300
Shrink_Allow	1.707
Req_Grey_Fabric	17412.189
Req_Beam_Length	1912.094
Weft_Count	1.289
EPI	1.33
PPI	1.353

Table 3

In order to check for multicollinearity within our data, we calculated the Variance Inflation Factor (VIF) for each of the predictor variables (Table 3). This metric reveals how strongly correlated predictor variables are. We calculated these values based on the MLR model for predicting Production; however, these values are virtually unchanged when using the MLR model for predicting Rejection.

According to Pennsylvania State University, "The general rule of thumb is that VIFs exceeding 4 warrant further investigation, while VIFs exceeding 10 are signs of serious multicollinearity requiring correction." As shown in Table 3, the VIF values for Required Finish Fabrics, Required Grey Fabric, and Required Beam Length are all incredibly high. There are a few different ways to account for this. One way would be to remove two of the three variables, in effect removing their collinearity effect. We, however, didn't want to lose the information that these variables provided to our models. Thus, instead of removing them, regularization techniques were used instead.

Beginning with Ridge Regression, which shrinks the coefficients but does not reduce them to zero, we must perform hyperparameter tuning to find the optimal value of lambda, the shrinkage parameter. We used 10-fold cross-validation to do this. Figure IV shows plots of cross-validated MSE vs. different possible values of lambda. For both Total Production and Rejection, we

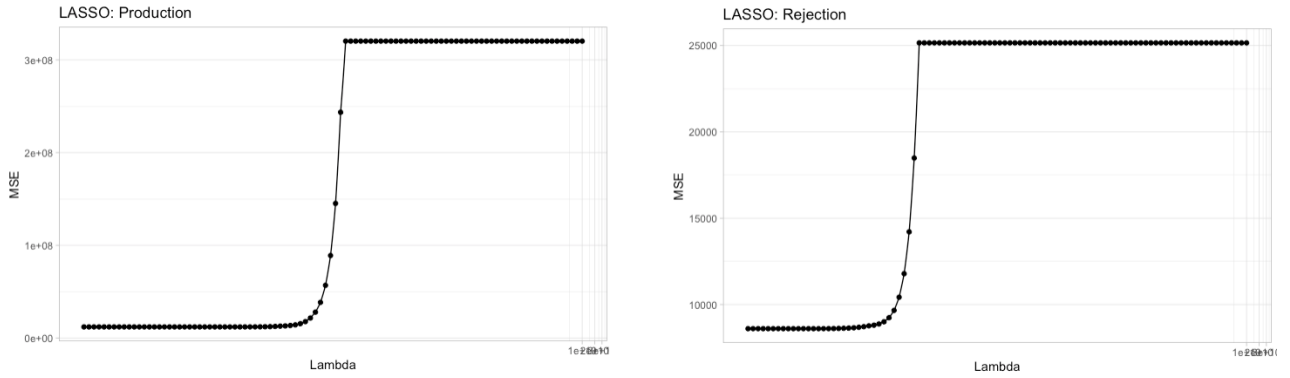

Figure V: Outcome Variable Distributions

observed very similar outcomes and found the optimal value of lambda to be very small, at 0.01. Despite the regularization, the ridge regression model failed to perform better than simple linear regression in terms of predicting Production. For predicting Rejection, it performed around the same. These results can be viewed in Table 4.

Method	Production RMSE	Rejection RMSE
Ridge Regression	3760.437 I 0.205	107.482 I 0.572

Table 4

After performing Ridge Regression, we shifted our focus on to LASSO. This method *does* perform feature selection by shrinking some estimates to zero, unlike ridge regression. To perform this method, we utilized the same 10-fold cross-validation technique as before in order to determine the optimal value of lambda for predicting both Production and Rejection. In Figure V, we again see plots of cross-validated MSE against possible values of lambda.

For predicting Production, we found the optimal value of lambda is much higher at 43.29. For predicting Rejection, however, the optimal value of lambda was again small at 0.01. Fitting these new models to our testing dataset and obtaining the new RMSE values, we see in Table 5 below that LASSO performed better when predicting production, and also performed very slightly better when predicting rejection.

Method	Production RMSE	Rejection RMSE
LASSO	3513.353 I 0.192	104.459 I 0.556

Table 5

To end our usage of linear models, we decided upon using a model that is a natural extension of a linear model but one that also allows for some nonlinear relationships — a Generalized Additive Model (GAM). In general, GAMs take the form:

$$g(E[y]) = \beta_0 + \Sigma_{i=1}^{m} f_i(x_i) + \epsilon_i$$

where each x_i represents a vector containing each predictor and each f_i is a function that maps the input vector to the output. These functions are both smooth and nonparametric, meaning that the functions are continuous on their domain and are entirely determined from the data, therefore no assumptions are made about the shape of the function before it is fit. Once the functions have been fit, their outputs are added together and then put through a link function, g, to obtain an output (Larson, 2015).

Because each function is fit independently and on a single feature of our dataset, visualization of these func-

tions allows us to get a glimpse into the inner-workings of the model to see the relationships it's forming — something we can't do with many other methods.

Before we discuss results, we must discuss what the plots of these functions — called Partial Dependence Plots (PDPs) — actually show. Because the model is additive, we know that each input will have some effect on the output that is independent of all the other inputs. Thus, each PDP will show the relationship between each individual predictor and the response variable and will represent the relationship that the model sees when all other variables are held constant (at their average value). If we see on the plot that y increases as x increases, we know that the model is seeing a positive correlation (Larson, 2015).

With this background done, we can move on to our approach to this method. We decided to start simply by fitting a GAM without performing any hyperparameter tuning to see if this method would be an improvement upon the other methods that we have proposed thus far. The results of the two models we fit, one to predict Production and one to predict Rejection, are shown in the table below.

Method	Production RMSE	Rejection RMSE
Unoptimized GAM	3214.998 | 0.17	68.164 | 0.438

Table 6

When compared to the results of the previous model, it is clear to see that allowing for some nonlinear relationships improved the accuracy of each model. The model that predicts Rejection is still not as accurate as the model that predicts Production — even when comparing the results to the Linear Regression model fit to start with — but, regardless, it has seen improvement.

Shown below, in Figures VI and VII, are the PDPs for each of the ten predictors for each of the two unoptimized models — the first set of plots belonging to the Production model and the second set belonging to the Rejection model.

There are a few things we wish to point out from these plots that have caught our attention. First, looking at the PDPs for the Production model, it can be seen that there are five variables that appear to have a roughly linear relationship with Production. This seemingly aligns with the fact that Linear Regression and the other linear models did quite well in predicting this outcome variable.

From the second set of PDPs, the one that corresponds to the Rejection model, we can see that fewer of the variables have such a linear relationship with the outcome of interest and, overall, the lines appear to be more "squiggly". One more thing we wish to note about these plots is that some of the variables seem to have opposite relations to each of the predictors. For example, Fabric Allowance tends to increase then decrease as Production increases but tends to decrease then increase as Rejection increases. A similar pattern can be seen in many of the plots shown.

Despite the fact that these GAMs performed quite well, we still wanted to see if we could push improvement even further. To do this, we went back and tuned each GAM in terms of its function's lambda values. Without tuning, each function had a lambda value of 0.6 (where lambda represents how much the function is penalized being nonlinear) (Oberhauser, 2017). Through some initial testing, we found that each model performed better when lambda was small (nonlinearity was penalized only slightly). We also found that, while it is possible to tune the parameter that corresponds to the number of splines that are used in estimating the

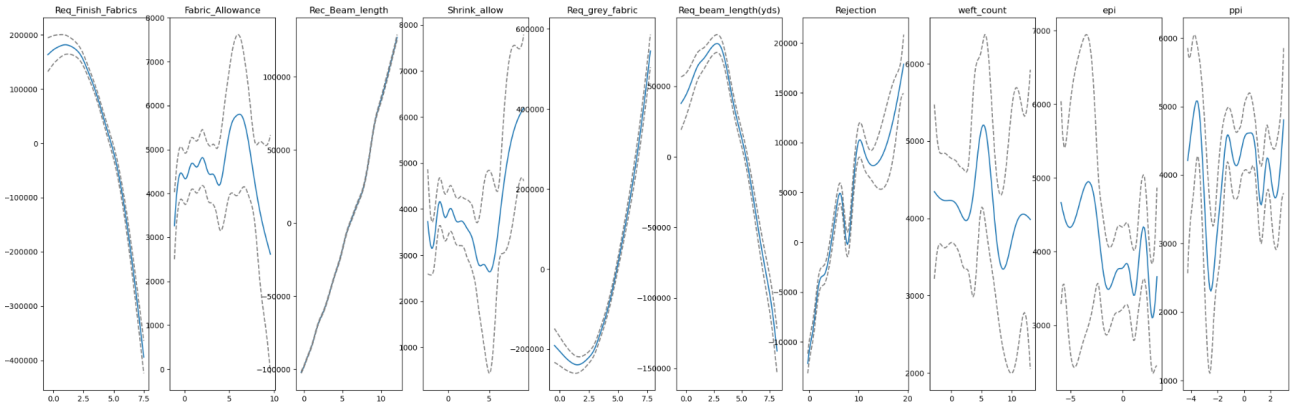

Figure VI

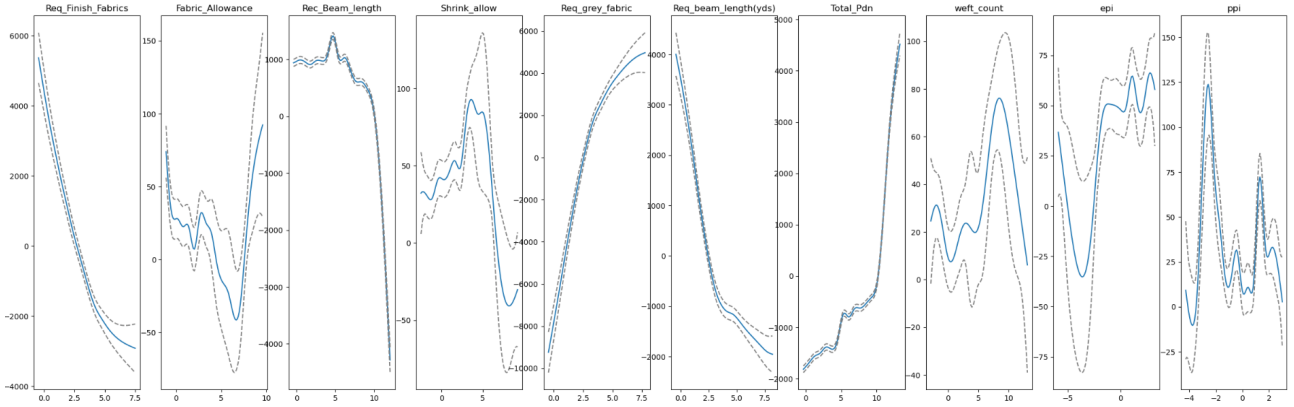

Figure VII

function, this parameter did not have as much of an impact on the overall model's accuracy as lambda did. Because adding more variables to tune would exponentially increase the amount of time it took to search for the best combination of parameters, we decided to leave this value at its default — 20 splines.

With this in mind, we used pyGAM's built-in grid search feature to find the best combination of three fairly small lambda values. Each GAM was fit 59,049 times — each time with a different combination of lambda values of 0.000001, 0.001, and 1 for each of the ten predictor variables ($3^{10} = 59,049$). This was no small task for pyGAM, and ended up taking just over 15 hours for the Production model and just over one day and eight hours for the Rejection model.

From previous testing, we knew that fitting the model on Rejection typically takes longer, so, we ran the code needed to fit the model outside of the Jupyter Notebook.

Before we show these plots, however, we wish to discuss the results of the new-and-improved GAMs that were fit. The results of these models can be seen in the table below.

Method	Production RMSE	Rejection RMSE
Tuned GAM	3067.677 | 0.162	66.027 | 0.424

Table 7

Clearly, tuning lambda had a positive effect on the accuracy of our model. Even though they took long to train, the results were worth the wait.

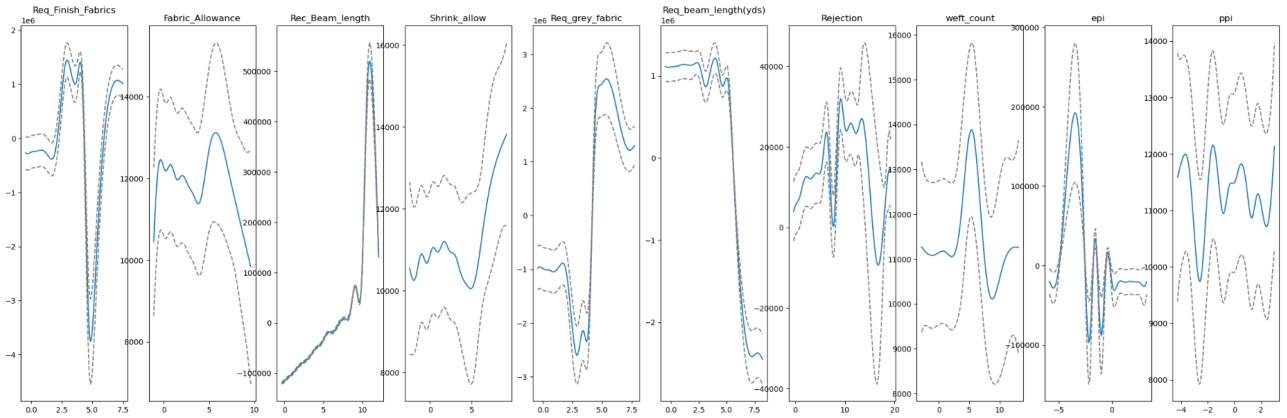

Figure VIII

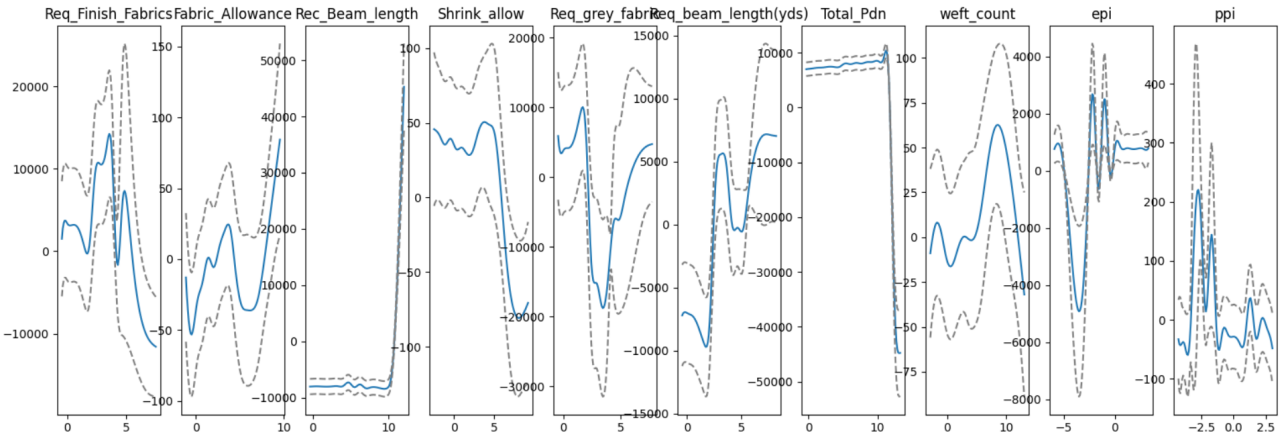

Figure IX

Finally, we can discuss the related PDPs for each of the models. Because we tuned lambda to potentially be much smaller than before, we expected the functions to be much more nonlinear than they were previously. For reference, the Production model had lambda values, in order, of 0.000001, 1.0, 0.000001. 1.0, 0.000001, 0.000001. 0.001, 1.0, 0.000001, and 1.0 and the Rejection model had lambda values, in order, of 0.000001, 1.0, 0.000001, 1.0, 0.000001, 0.000001, 0.000001, 1.0, 0.000001, and 0.01. The PDPs for each of these models can be seen in Figures VIII and IX above.

Analyzing the PDPs corresponding to the Production model, we saw that, though many of the relationships are still quite linear, within those linear relationships exist more "squiggle". This is unsurprising, however, as many of the optimal lambda values were quite small, meaning that nonlinearity was not penalized.

Next, moving on to the PDPs for the Rejection model, we observed results similar to what we expected given that most of the lambda values for this model were quite small. Most of the functions are much more "squiggly" and there are fewer linear relationships between the predictors and Rejection. It seems that a model predicting Rejection greatly benefits from allowing nonlinearity.

What we wish to note about this model is that, as was seen when moving from the linear models to GAMs, adding in nonlinearity improved the predictive accuracy of Rejection more than it did the predictive accuracy of Production. To push this idea to the limits,

we decided upon using one more method — a highly non-linear one.

This final method was a Neural Network. Neither of us have much experience in training Neural Networks, so we took this opportunity to a) try them out and b) test to see whether or not adding in more nonlinerity would have the assumed effects (improving Rejection RMSE more than Production RMSE). To slightly complicate things further and in order to work with a method we had never worked with even in passing, we decided to train two Bayesian Neural Networks (BNNs) to predict each of our outcome variables.

This approach to a Neural Network allows for some difference in the actual predictions that it makes. Simply put, if we put one number through the BNN, the prediction it makes for that single input would be different every time. Because of this, we knew we needed to change our approach to estimating the RMSE of the model. Previously, we could directly take this measurement based on the results of the model, but, because predictions vary, we decided to take the mean of 1,000 predictions and take that as the model's output. This was good, but we knew we could do better.

In order to do better, we took into account the standard deviation of the predictions. Given we calculated 1,000 different estimates, we could calculate their standard deviation. Next, we could add and subtract it to each mean estimate to get a better estimate of an upper and lower bound estimate of RMSE.

But, before we discuss the results, we wish to discuss some of the work that went into obtaining the final model. First, there were a few parameters to tune in order to change our results slightly. The first choice was that of which activation function to use. After some research, we saw that ReLU was the clear favorite among data scientists, though there are many different activation functions to choose from. We ran a few trials with a few different activation functions and found that LeakyReLU, an improved version of ReLU, garnered the best results (though the margin was very small). The downside to using LeakyReLu, as we quickly discovered, was that using this activation function made the model take longer to train.

The next parameter to tune was the learning rate. We started with a mid-sized learning rate of 0.01 but, through trial and error, ended up using a learning rate of 0.001. We had also tried 0.1 but found that, expectedly, larger learning rates led to larger RMSEs. Unfortunately, as we decreased the learning rate, the BNNs took longer to train.

The final place of tuning came with the number of epochs to use. As a general rule of thumb, more epochs tend to lead to better performance but also longer training times. After trying different values, we ended up satisfied with the training time and RMSE of 4,000 epochs. Below are the results of the final Bayesian Neural Networks that were trained.

Method	Production RMSE	Rejection RMSE
BNN	3074.998 | 0.162	62.417 | 0.401

Table 8

The upper and lower bounds for the Production model were both 0.162 and these bounds for the Rejection model were 0.399 and 0.412 (in standardized units). It can be seen that this non-linear method was useful in improving the accuracy of the Rejection model, but did not improve the accuracy of the Production model.

Method	Production RMSE	Rejection RMSE
Linear Regression	3656.91 | 0.19	91.43 | 0.57
Ridge Regression	3760.44 | 0.21	107.48 | 0.57
LASSO	3513.35 | 0.19	104.46 | 0.56
GAM	3215.00 | 0.17	68.16 | 0.44
Tuned GAM	3067.68 | 0.16	66.03 | 0.42
Bayesian NN	3075.00 | 0.16	62.42 | 0.40

Table 9

Conclusions:

The results from all of the different models fit have been compiled and are shown in Table 9 above for reference.

From these results, it seems as though Total Production is an easier outcome to predict and can be done quite simply by using a Multiple Linear Regression model. Though we were able to reduce the error slightly by trying more complicated models and introducing nonlinear components, we see no need to introduce more complexity and longer run times by using more complex modeling techniques.

Though we have limited knowledge of how textile weaving factories operate, it seems reasonable to say that the total amount of fabric produced in the factory at a given time is a (roughly) linear product of the other factors that go into production. For example, if the amount of required fabric is higher, it seems to follow that there will be a linearly higher amount of fabric produced. Thus, it makes sense to us that, even though some of the assumptions of linear regression were violated, we were able to achieve such low RMSE values even with the simplest of linear models.

We initially stated we had no prior knowledge or intuition about the underlying relationship between the predictors and Rejection, but after modeling these relationships, we believe that they are highly nonlinear and require nonlinear methods in order to accurately predict. As can be seen in Table 9, the largest improvement in RMSE in the Rejection models occurred when we introduced this element of nonlinearity with the unoptimized GAM.

Once nonlinearity had been introduced, we were unable to improve RMSE much further, but, interestingly, when predicting Rejection, we saw more of an improvement in RMSE from the unoptimized GAM to the BNN than we saw going from Linear Regression to the BNN when predicting Production (0.04 vs. 0.03 in standardized units).

Because of these results, we believe that devoting more time to train better models that allow for nonlinearity will prove to drive the RMSE of models that aim to predict Rejection down even further. We also believe that such methods may be useful in attaining lower RMSE values for models that aim to predict Production, but the time, effort, and resources needed to create such a model may not prove to be worth investing in given the results of such simple models.

REFERENCES:

[1] Ahmed, T., & Uddin, S. (2023). Textile weaving dataset for machine learning to predict rejection and production of a weaving factory. *Data in brief*, 47, 108995. https://doi.org/10.1016/j.dib.2023.108995

[2] *A Tour of pyGAM.* (n.d.). https://pygam.readthe docs.io/en/latest/notebooks/tour_of_pygam.html

[3] Baheti, P. (2021). *Activation Functions in Neural Networks [12 Types & Use cases].* V7. https://www.v7 labs.com/blog/neural-networks-activation-functions

[4] Chatterjee S, Hadi AS. Regression Analysis by Example. 4th ed. Hoboken, New Jersey: Willey-Interscience; 2006:375.

[5] *Detecting Multicollinearity Using Variance Inflation Factor.* (n.d.). The Pennsylvania State University. (2018). https://online.stat.psu.edu/stat462/node/180/

[6] Hassanien, A. E., Abouelmagd, L. M., Mahmoud, A. S., & Darwish, A. (2023). An optimized backpropagation neural network models for the prediction of nanomaterials concentration for purification industrial wastewater. *Engineering Applications of Artificial Intelligence*, 126, 107010. https://doi.org/10.1016/j.eng appai.2023.107010

[7] Larsen, K. (2015). *GAM: The Predictive Modeling Silver Bullet. MultiThreaded.* https://multithreaded.sti tchfix.com/blog/2015/07/30/gam/?source=post_page—457df5b4705f————————

[8] Murel, J., & Kavlakoglu, E. (2023, November 21). *What is ridge regression?.* IBM. https://www.ibm.com/topics/ridge-regression#: :text=It%20corrects%20for%20overfitting%20on,by%20overfitting%20on%20training%20data.

[9] Oberhauser, P. (2017). pyGAM: Getting started with generalized additive models in Python. *Medium.* https://codeburst.io/pygam-getting-started-with-generalized-additive-models-in-python-457df5b4705f

[10] Paialunga, P. (2022). *From Theory to Practice with Bayesian Neural Network, Using Python.* Toward Data Science. https://towardsdatascience.com/from-theory-to-practice-with-bayesian-neural-network-using-python-9262b611b825

[11] Pereira, F., Pinto, L., Soares, F., Vasconcelos, R., Machado, J., & Carvalho, V. (2024). Online yarn hairiness- Loop & protruding fibers dataset. *Data in brief*, 54, 110355. https://doi.org/10.1016/j.dib.2024.110355