Predicting Manufacturing Defects for 3-D printed parts

I. Introduction: problem statement and goals.

Large manufacturers typically have dozens if not hundreds of vendors supplying parts for further assembly or operation. While many parts may be fungible, or easily replaced if defective, custom 3D printed parts are less so, and high defect rates can lead to lost work, cost overruns and production delays. But it is hard to predict whether a part or a supplier will be more prone to high defects, or not; it is especially hard when information on the particular operations, methods and so forth used by a supplier are unavailable.

In this analysis a large number of different part shipments are analyzed along with many vendor-provided characteristics that relate, broadly, to production efficiency and quality. The goal is to identify, if possible, whether these factors can be used to predict the likelihood of higher manufacturing defects, so that the process of vendor qualification and selection can be improved. In many manufacturing environments, especially where 6 sigma practices are employed to keep quality high, vendors undergo qualification before receiving orders and contracts, to establish their reliability, competitiveness, quality and so forth. But defect rates aren't always disclosed and may not be factored into such ratings, especially when producers are making custom 3D printed parts.

We will analyze a large dataset of over 3200 part shipments and build a model to predict high (>3 defects/thousand) defect rates. This report details the data fields and their characteristics, the data processing, model assumptions, results and finally provides recommendations for users. The model may be used as part of vendor scoring or qualification systems for manufacturing operations and purchasing departments.

IIa. Initial overview of the dataset

This report is based on the Kaggle "Predicting Manufacturing Defects Dataset"[1]. The data consists of 3240 shipments, with 17 metrics crucial for predicting high or low defect occurrences in production processes. These metrics are somewhat general and involve production levels, ratings, safety, efficiency and supply chain information. A detailed examination of the variables follows. The dataset itself was remarkably free of missing data fields, duplicates, data type errors, inconsistencies, etc. and did not require any pre-processing before the assessment of these data fields. The dataset provides a 'Supplier Quality' rating, as well as a 'Quality Score', however, it is unclear what the sources and methods were for these ratings.

---

[1] **Dataset Usage and Attribution Notice:** This dataset, shared by Rabie El Kharoua, is original and is made available under the CC BY 4.0 license.

IIb. Overview of dataset variables

In Table 1, below, the various metrics and their characteristics are summarized.

Metric	Range	Data type	Comments
Production Volume	100-1000 units/day	Integer/binned categorical	relatively uniform (but noisy)
Production Cost	$5000-20000	Cost per day	
Supplier Quality	80%-100%	Continuous	Rating criteria unknown—excluded from model
Quality Score	60-100%	Continuous	Rating criteria unknown ("overall quality assessment")
Maintenance Hours	0-24 hours/week	Integer (categorical)	Relatively uniform
Stockout Rate	0-10%	Continuous	
Inventory Turnover	2-10 (ratio of turnover)	Continuous	
Worker Productivity	80-100%	Continuous	(low predictive value; excluded from model)
Safety Incidents	0-10 incidents (per month)	Integer (categorical)	Somewhat bimodal, per fig. 1
Energy Consumption	1000-5000 kWh	Continuous	
Energy Efficiency	0.1 to .5	Continuous	
Additive Process Time	1-10 hours	Continuous	(low predictive value; excluded from model)
Additive Material Cost	$100-500	Continuous	
Delivery Delay	0-5 days	Integer (categorical)	
Downtime Percentage	0-5%	Continuous	
Defect Status			Author-supplied labels (not used in the model)
Defect Rate	0.5 to 5 defects/thousand parts	Continuous; later binned into high/low at high > 3/ thousand	Target variable

Table 1. Defect Rate and potentially associated metrics, with brief descriptions of each.

Overall defect instances and dataset balance

Per the author, "the dataset focuses on defect instances more because they do not occur often. However, non-defect instances were added too; for this reason, the dataset is imbalanced." Based on this report we found about 44% of records had a high defect rate (1439 out of 3240 observations), where we label 'high defects' as > 3 defects per thousand parts (which seems to accord with the author's labeling). Hence the dataset is somewhat imbalanced, which was taken into account during modeling and analysis of results.

IIc. Variable distributions

Of interest is the question of whether any of these variables, on their own, exhibit unusual features which may later play a role in the model. While the data ranges and some comments on each are presented in Table 1, above, we present a few of the variables that stood out as having somewhat unusual distributions. (Frequency graphs for some of the other variables are provided in the Appendix). Note that some data fields are integer-valued, and in other cases continuous; where appropriate, we present either raw or binned data to demonstrate trends.

In Fig. 1 below we present histograms for maintenance hours and safety incidents. Maintenance hours has a slight peak at 15 hours per week, but is otherwise relatively flat: the number of low-maintenance records isn't far off from that of high-maintenance. Most of the data were similarly flat; an exception is Safety Incidents, which broadly break down into 'very safe/no incident' suppliers, and 'unsafe' suppliers with 8 or more incidents per month, with everyone else uniformly spread in between.

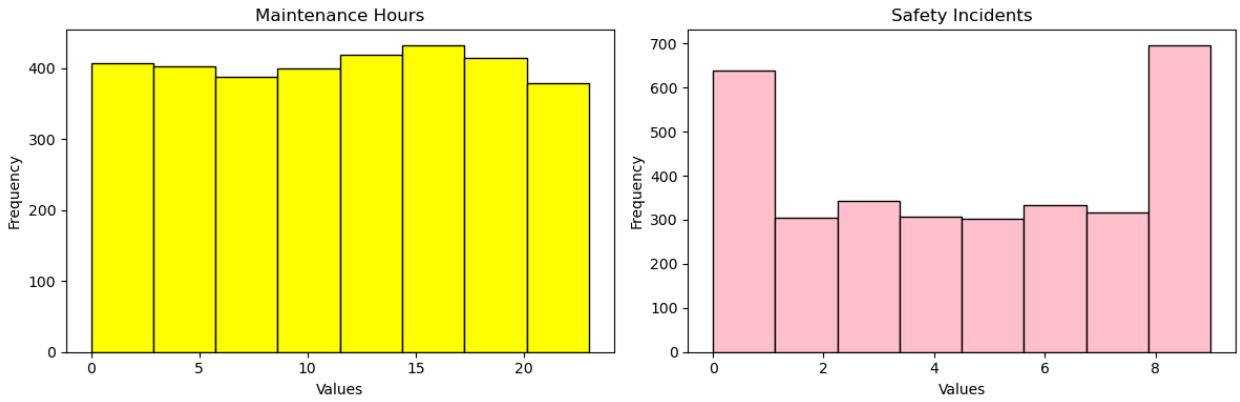


Figure 1. Frequency graphs of suppliers in the data set with maintenance hours, and safety incidents. Note the relative uniformity of the former and bi-modality of the latter.

Because it is difficult to view all of these at once, Figure 2 provides box plots of four other variables: Supplier Quality, Defect Rate, Quality Score and Downtime Percentage. The distributions do not appear significantly skewed, nor are there outliers to contend with.

The same pattern is present in the remaining variables. For modeling this will mean less work to deal with skew or the presence of outliers; fortunately, no data had to be removed for the analysis to proceed.

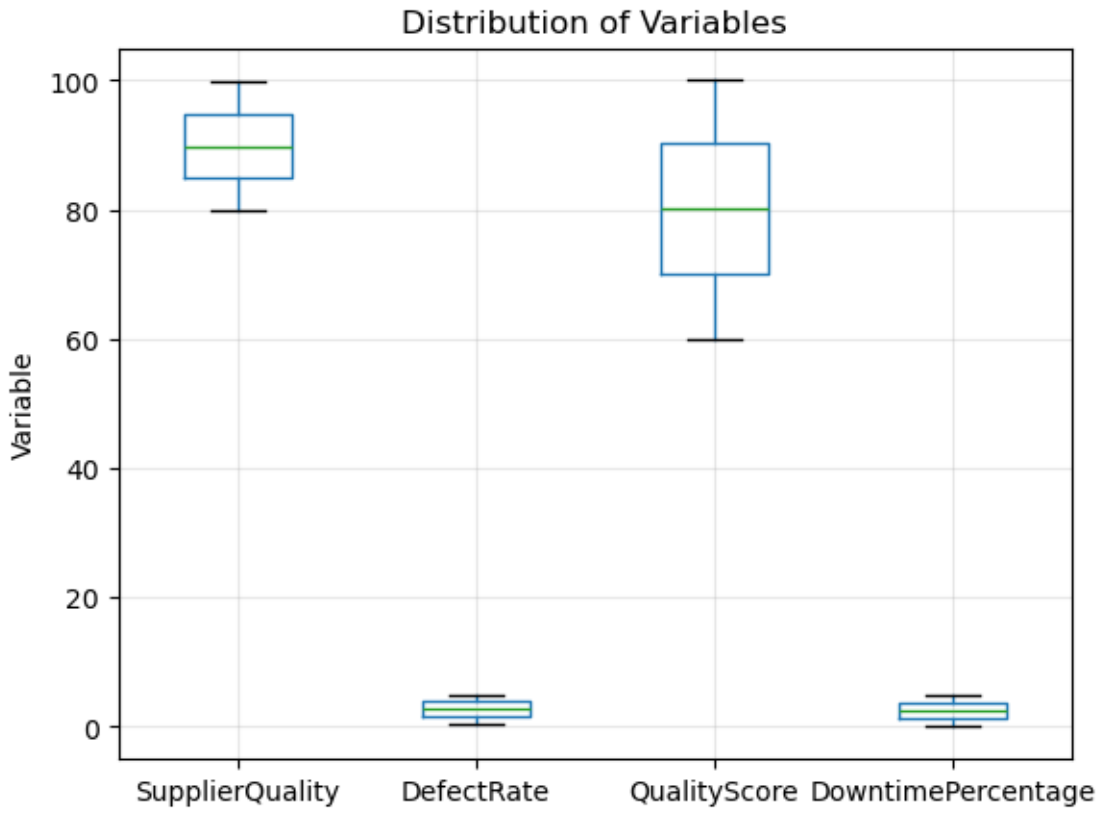


Figure 2. Box plots showing the distribution of several features: note the symmetry and lack of outliers.

III. Exploratory Data Analysis

We now examine the relationship between the variables with measured defect rates. The first step is to examine correlations to determine which variables may be particularly useful in building a model. In figure 3, we compare the variables with each other (excluding production volume, defect status, and maintenance hours, which are dealt with later). The correlations were all less than 4% (light to dark yellow, in the figure). When comparing each input variable to Defect Rate, which is our target—the correlation plots are very noisy.

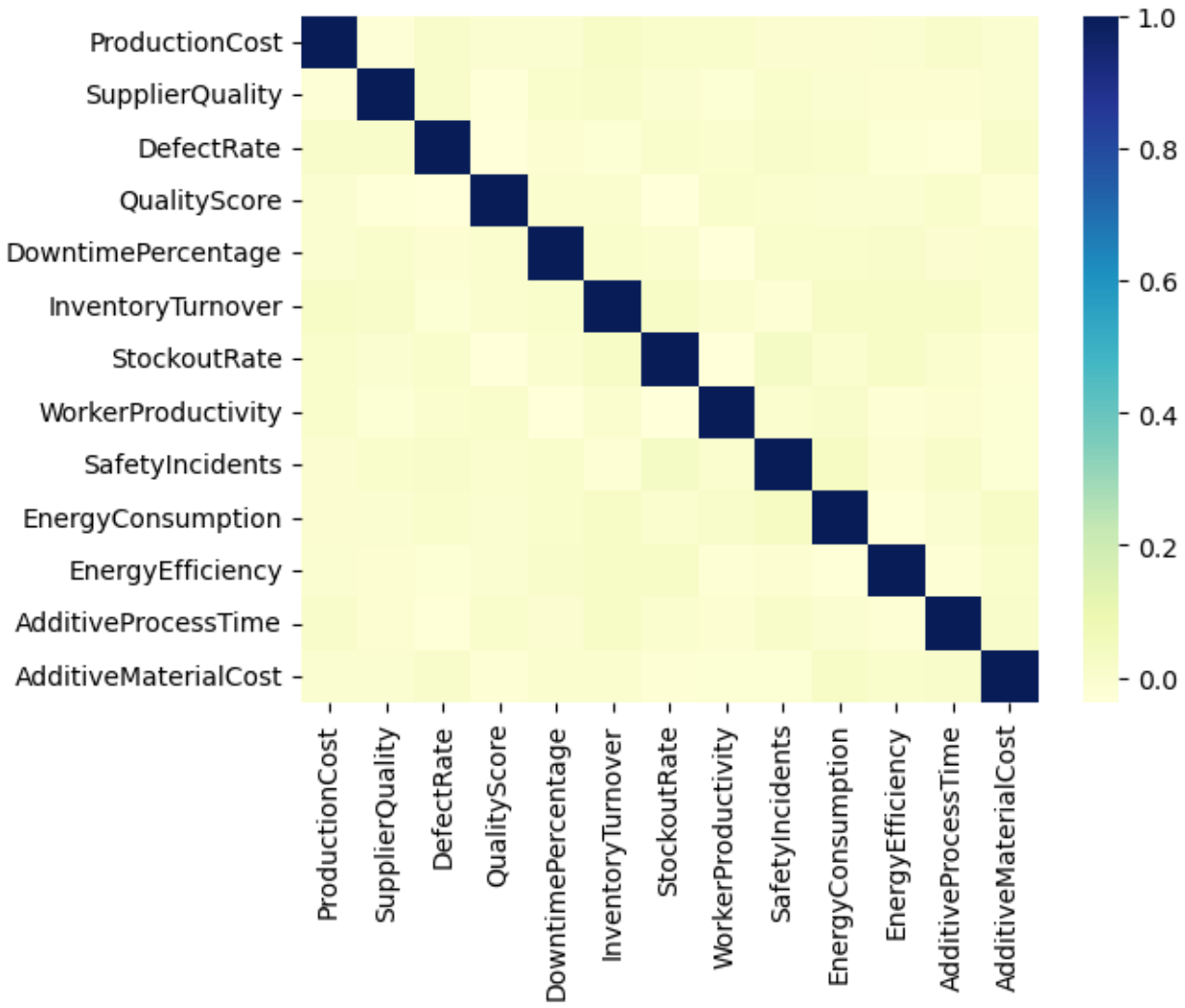


Figure 3. Correlation heat map of most variables in the dataset. In general, the correlations are very low and are likely to be weak predictors of the target (defect rate).

An example of the noisy relationships is shown in Figure 4, in which the quality score is depicted in a scatter plot against the defect rate. It is almost impossible to see a pattern. The same was true for the other variables (see also the Appendix).

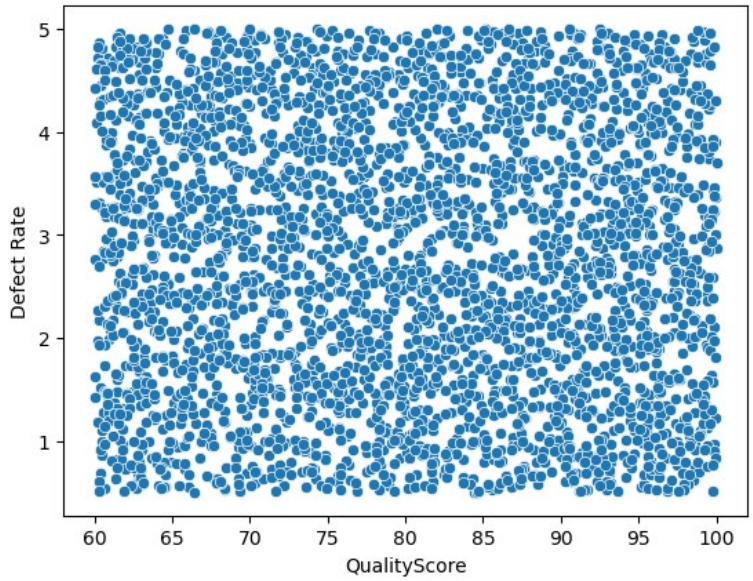


Figure 4. Scatterplot of Quality Score and Defect rate: the noise makes any apparent correlation barely discernable.

Under the circumstances, the decision was made to switch Defect Rate to a categorical variable, rather than continuous. The authors of the dataset appear to label most shipments with more than 3 defects/thousand 'high' in their 'defect status' column, so we used this threshold and converted defect rate to 'high' and 'low' accordingly.

III. EDA, part 2.

With defect rate now converted to a categorical target variable, the trends between variables and the target were revisited. In this step, production volume was also binned to create a categorical variable (using bins of width 100 ranging from 100-1000). The decision to bin production volume was based on the initial analysis of its distribution, which was noisy and poorly correlated to defect rate (see also the Appendix).

A typical comparison between high and low defect rates is shown in Figure 5, which depicts downtime percentage, below. To interpret these plots, it helps to look at the trend between low and high defect rates (orange and blue). If the trends track each other, the variable is less informative than if the patterns are distinct. In all plots the total number of 'high defect' lots is smaller in general, which simply reflects the overall population of the dataset.

For example, as downtime percentage increases, high defect rate shipments seem to trend downward (although the opposite is true for low defects). It is tempting to reason that more downtime means more efforts are being made to correct for machines drifting out of calibration.

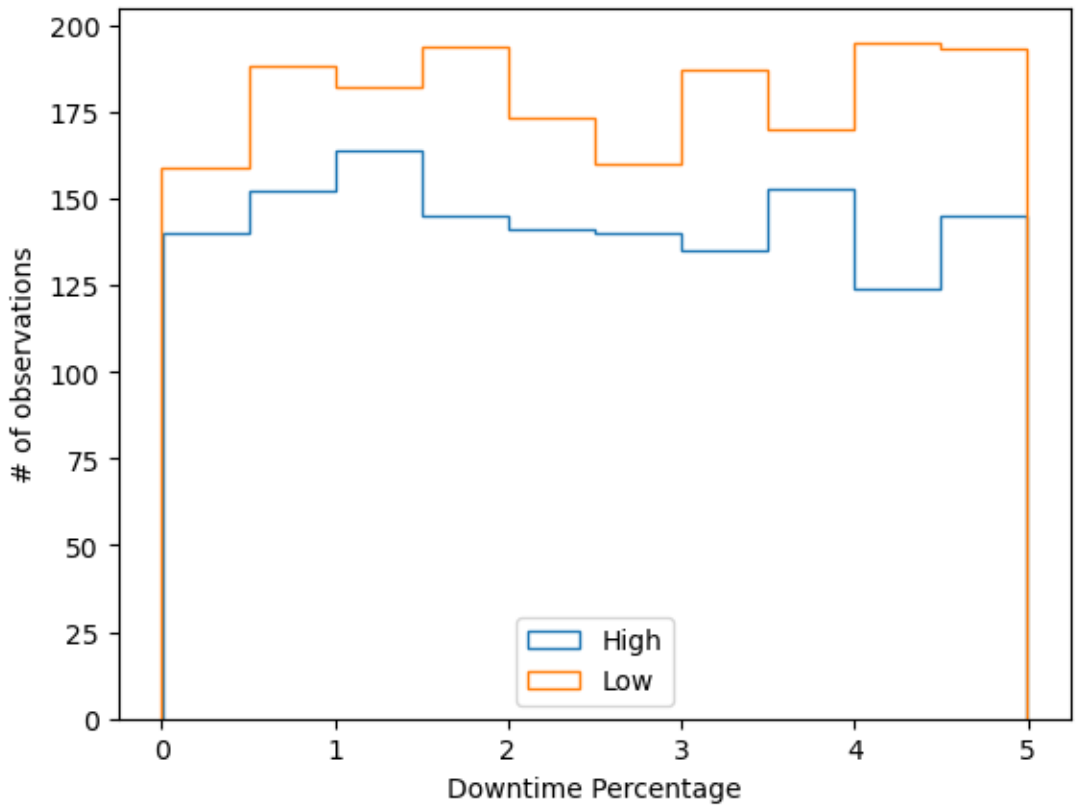


Figure 5. Downtime percentage seems to improve outcomes for defect rate, as defect rate increases: a higher percentage appears to lead to fewer high-defect and more low-defect shipments.

In terms of feature selection, some variables were ultimately set aside because they appear to be mostly noise, in particular: supplier quality, worker productivity, and additive process time. Wherever some pattern, however weak, was discernable, the variable was kept.

It was very surprising to see that 'supplier quality' was effectively silent about defect rates! This underscores that it is not clear how these ratings were generated: what were the criteria for high versus low supplier quality? It also underscores that a model which augments a supplier score on other dimensions can be improved by adding some information about probable defect rates. As a final observation, the ratings for Supplier Quality all ranged from 80 to 100 %, which gives the impression that, overall, these suppliers were relatively good to begin with (so why would well-rated suppliers be expected to frequently deliver high-defect shipments?).

Graphs for some of the other variables, comparing high and low defect rates, are included in the Appendix.

IV. Preprocessing and Training Development.

The prior analysis led to a revised (binary) target and the rejection of some variables for the final models. As noted in Figure 2, and elsewhere, the continuous variables had different means and ranges. These had to be standardized so that comparisons could be made on an equal footing; in other words, variables would not improperly receive distorted weights depending on different means. The python 'StandardScaler' was used to transform the continuous variables so that the mean of each was zero and the standard deviation is one. (The scaler does this one variable at a time. It does not compute an overall mean or standard deviation). Earlier we saw very variables with unusual distributions; if anything, deviations from a uniform distribution were relatively mild and somewhat uncommon.

A different approach was used to treat the integer-valued variables, such as delivery delay, maintenance hours, safety incidents, and binned production volume. For these, dummy variables were created, which act as 'categories' for the purpose of the model.

Finally the data were randomly split into train and test sets, in which 80% of the data were assigned to the 'train' category and 20% into 'test'. The defect status (high/low) was separated as the target variable for training and test. Unused columns were discarded, leaving 3240 observations in total, with 58 variables.

V. Model development

To classify the data as having either high or low defects, three types of models were built: logistic regression, decision trees/random forests, and support vector machines. Each was first used in some simple tests, then tuned for best performance, and compared using various test metrics. In general, the models performed better than random guessing, however, the low-correlation variables limited the quality of the final models, which had only modest predictive gains overall. Ultimately, a boosted random forest regressor was found to have the best overall performance.

The key performance metrics for binary classification include accuracy (percent of correct classification/all data), precision (true positives/true positives and false positives) and recall (true positives/true and false negatives). Before discussing these in more detail, it is important to keep in mind that this dataset is only slightly imbalanced—which means it's important to look at overall performance rather than picking one metric over another.

For clarity, the model considers a positive to be low defect rate (low defects are desirable).

Va. Logistic regression models.

As an initial trial run, a logistic regression model was run to get a baseline on performance, using a guess at model parameters (with a high number of iterations for convergence, and a high number for 'regularization', a parameter used to avoid overfitting, discussed in more detail later). Each of these is calculated for both the training and test data; for now, we will consider accuracy but at certain points we will return to these metrics and analyze how best to use them with different models.

If a purely random model (coin toss) were used, we'd expect an accuracy of 50%. Rather discouragingly, the initial model showed an accuracy of 53% on the test data. That is, the model is better than random guessing but only modestly so. However, recall that the input variables are very noisy and generally do not appear to have direct causal relationships to the underlying processes, which actually drive defect rates on the manufacturing floor.

Before moving forward, cross validation was performed to ensure that the random train/test split wasn't biased somehow—by checking with multiple folds of data we can be more confident in the reliability of the results. Running cross validation on the logistic regression, we used the f1 score (which balances both precision and recall) and got .49 across the folds with a standard deviation of .01. This means there isn't a lot of variation between the folds, and we can be more confident in the overall model results.

Next we performed a grid search of different values of the regularization parameter, C, along with different optimization approaches (ridge vs lasso). The search returned best parameters for logistic regression different from what we started with (optimal C was 100, and L1, or lasso, was selected). That said, the optimizer parameters returned the same overall accuracy of 53%. A summary of the optimized results along with the confusion matrix (which summarizes how many results fell into true and false positive/true and false negatives) are presented in Table 2 and Figure 6, below.

Classification Report for Test Data---Logistic Regression

	precision	recall	f1-score	comments
	0.57	0.76	0.65	low defect shipments
	0.42	0.23	0.30	high defect shipments
accuracy			**0.53**	Small model improvement
macro avg	0.49	0.49	0.47	
weighted avg	0.50	0.53	0.50	

Table 2. Summary of performance metrics for the final Logistic Regression model.

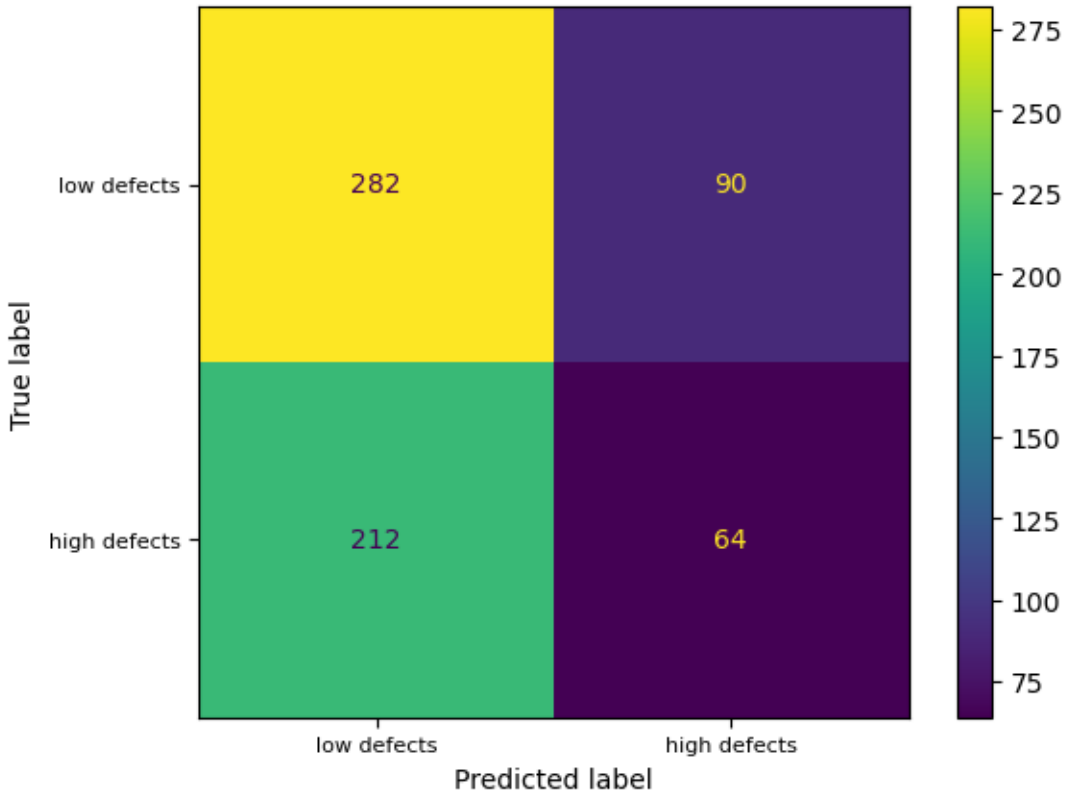


Figure 6. Confusion matrix for the final Logistic Regression model. Only 64 high-defect shipments were correctly found by the model (with 90 falsely labeled as high, and 212 incorrectly labeled low).

Vb. Modeling with Decision Trees and Random Forests

One of the advantages of decision trees as a model is the relative simplicity and interpretability, which certainly applies in this case, as will be seen in the discussion.

We began by allowing the model to build a tree without limit on depth; however, this approach led to a very overfit model with results barely better than random on the test set. Consequently the model was constrained to have a depth of 4; this led to better and more interesting results (accuracy of 56%). The tree is displayed in Figure 7, below:

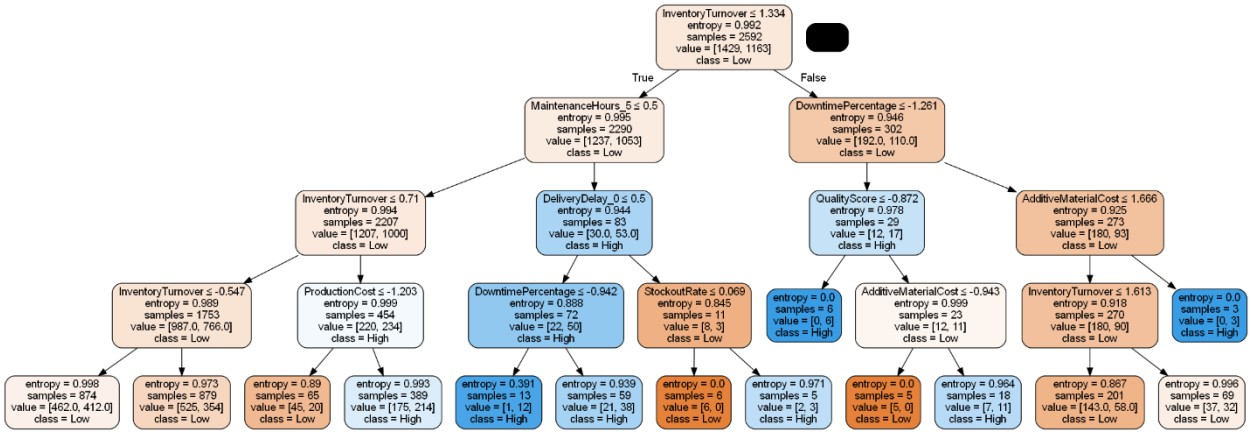


Figure 7. A decision tree with depth of 4. Note the features used by the model in splitting the categories.

This decision tree reveals interesting information about which variables are leading to a better split of the data. The tree is relying heavily on inventory turnover, maintenance hours, downtime percentage, delivery delay, and quality score. While other variables make an appearance, these few give insight into what may be happening behind the scenes.

Maintenance hours and downtime percentage (as we saw earlier in Figure 5) are open to some common-sense interpretation. In the model, we can see that maintenance hours are very slightly increasing for low and slightly decreasing for high defect shipments. Beyond that trend, it is not entirely clear why this might be: there is scheduled maintenance and unscheduled, when equipment breaks down; maintenance may be high or low depending on the type of printer and may have little to do with quality.

Downtime can also be scheduled or not, and may have to do with repairs or with, say, periods when the machinery is unused or during set-up between runs. More downtime, as we saw earlier, was associated with a greater split between high and low defect runs, suggesting that downtime might have to do with efforts to correct for machinery drifting out of calibration, more time spent during set-up to get things just right, etc. (Ambiguity

unfortunately muddles the picture: would recalibration be counted toward downtime, maintenance, or both?)

Delivery delays could go hand in hand with downtime percentage, to the extent that delays stem from production and not transport and logistics. The delays seen between high and low defect shipments seem to follow the same trend, though there appears to be a difference in proportion for long delays.

The quality rating also appears to be informative, so however it was derived, it very likely included some consideration of defect rates (in contrast to 'supplier quality', which apparently did not).

But the primary variable used by the model is inventory turnover (and to a lesser extent, the stockout rate, which one would expect to be related). This variable certainly has different trends for high and low defect shipments: higher inventory turnover seems to lead to better results (fewer high defect shipments). Perhaps materials flowing through the system at a high rate signals a higher overall quality for the manufacturer.

To a lesser degree we also see additive material cost playing a role; additional costs of this nature would be associated with re-work; these costs tend to rise for high defect shipments and are flatter or slightly decreasing for low defect shipments.

Other variables (e.g., energy efficiency, worker productivity) do not seem to be useful in predicting the target, which seems reasonable, as it is hard to establish a clear reason why these would play a direct role in defect rates.

While the results of building a single (shorter) tree were better (deeper trees were also tested without noticeable improvement), in the next section we will use random forests to address problems with noise and overfitting.

A summary of the decision tree performance is included in Table 3, below. Although the accuracy has improved, and overfitting has been reduced, there is an important caveat: how well is the model detecting high defect shipments, versus low? It turns out, it is not doing as well as the first model in finding the high-defect shipments. Accuracy, while useful, is not the only metric we need to examine. As we proceed, combined metrics such as the f1 score will be used in an attempt to balance the model and improve detection of both classes.

Decision tree model metrics - max depth 4

Accuracy:	54.3%
Balanced accuracy:	50.0%
Precision score for "High"	42.6%
Precision score for "Low"	57.4%
Recall score for "High"	21.0%
Recall score for "Low"	79.0%

Table 3. Model metrics for a decision tree with max depth of 4.

Vc. Random Forests (and modeling trade-offs).

The decision tree used earlier was relatively short, with a max depth of 4. However, further refinement is possible, and in this section, we analyze the use of many such trees and how they might collectively do a better job predicting our target variable.

Initial trials did not yield improvement, but a more systematic parameter search was helpful. As mentioned earlier, it became clear that these models had to be trained to optimize the f1_weighted score at this stage. Model optimization with accuracy was unhelpful for random forests: the models were performing extremely well finding low defect shipments, while missing nearly all the high defects! The f1 score (weighted, because our dataset is slightly imbalanced) corrects this behavior and results in better class labeling.

The consideration of model trade-offs now arises: What would we prefer the model is really good at, if its performance isn't going to be perfect? For some producers, it may be preferable not to have a high rate of false negatives--good shipments labeled as bad. Depending on the producer, perhaps it would be better to have a model which only picks a small percent of the 'bad apples' but does so with high accuracy. That said, models that simply label very few shipments as 'high defects' likely aren't very valuable, so we will favor those that, on balance, tend to do better detecting the high defect class.

Returning to the random forest, we began the parameter search. This was a randomized search with cross validation, in which the number of estimators, maximum tree depth, maximum features, minimal samples per leaf and minimum samples per split, and use of bootstrapping, were tested. The model found better performance for a high number of

estimators (n=1050), and a deep tree (depth of 40) with a small number of samples/leaf: this combination of parameters suggested the model might still be pushing toward overfit.

To counter this possibility, a further grid search was run to nudge the model toward better parameters. The grid search resulted in a shorter tree and fewer estimators (max depth of 10 and 900 estimators). Table 4 shows the metrics for the model, and Figure 8 shows the confusion matrix.

Best RF model per parameter search

Accuracy:	54.6%
Balanced accuracy:	49.0%
Precision score	38.75
Recall score	86.82

Table 4. The optimized random forest metrics. Accuracy is somewhat higher but balanced accuracy and precision are lower than hoped for.

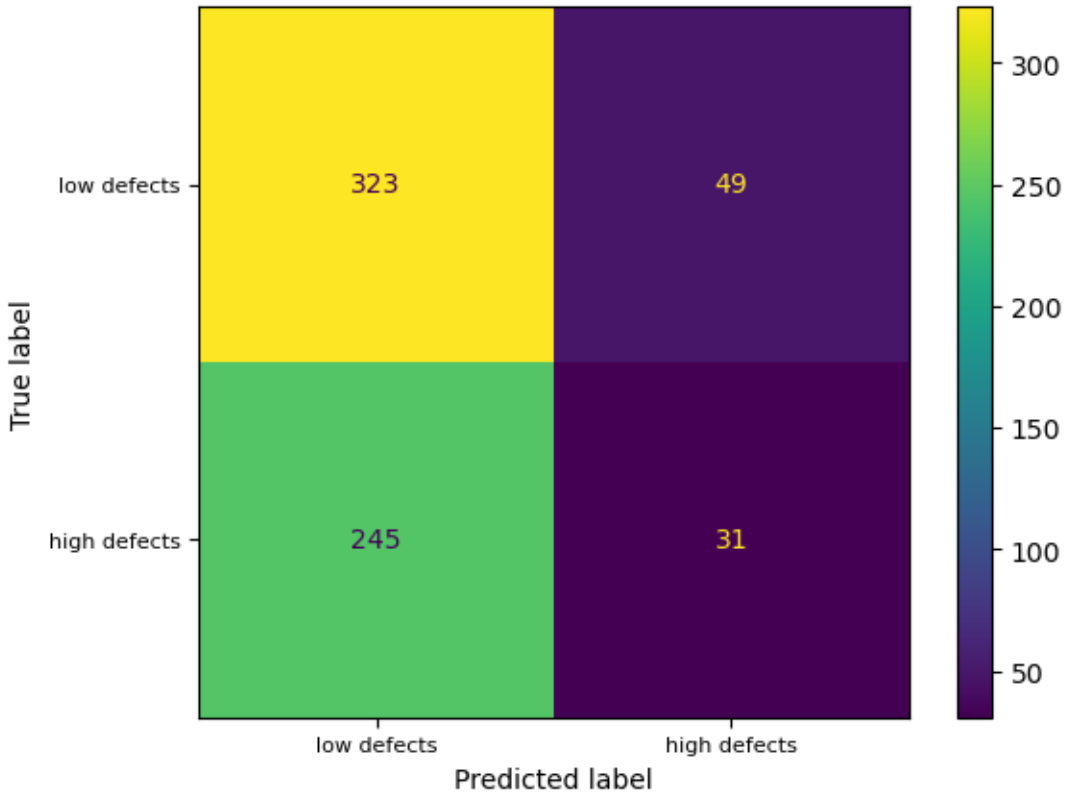


Figure 8. The optimized random forest model is still having trouble detecting the high defect classification, with only 11% correctly labeled as such.

Finally gradient boosting was applied to improve model performance. Initially random forests were used because of concerns about noisy data, which was evident from the beginning (see, e.g., Figure 4). Many decision trees were averaged (bagging) to try to reduce the impact of noise and prevent overfitting. However, the results were somewhat unsatisfactory from the standpoint of finding high defect shipments. Gradient boosting is sometimes chosen for unbalanced datasets where it is harder to find the positive class, so while this dataset wasn't ideally suited from that standpoint it was worth investigating—the models weren't doing great finding the high defect shipments.

After adjusting the learning rate for the boost model, the following results were obtained. A summary of model performance is shown in Table 5, below; the confusion matrix is presented in Figure 9. Now the model does a better job finding the high defect class, and represents an improvement over logistic regression.

Gradient boosted model metrics

	precision	recall	f1-score
Low defects	0.58	0.61	0.59
High defects	0.43	0.39	0.41
accuracy	0.52		
macro avg	0.50	0.50	0.50
weighted avg	0.51	0.52	0.52

Table 5. Summary of model performance for gradient boosted random forest model. Although the model is more balanced, the overall performance is still rather low.

The drawback to all these efforts is that overall accuracy remains not especially high; the accuracy of the various models that have been tried has not risen much above the low 50's, percentagewise, and when it does obtain higher accuracy this comes at the cost of poor detection of high-defect shipments.

With the foregoing in mind, we next turn our efforts to another model class, support vector machines, to see whether a different approach can better separate the two classes. In all of this it is still important to recall the noisy inputs: only so much can be reasonably expected from these models when the inputs are as poor as they are on signal/noise.

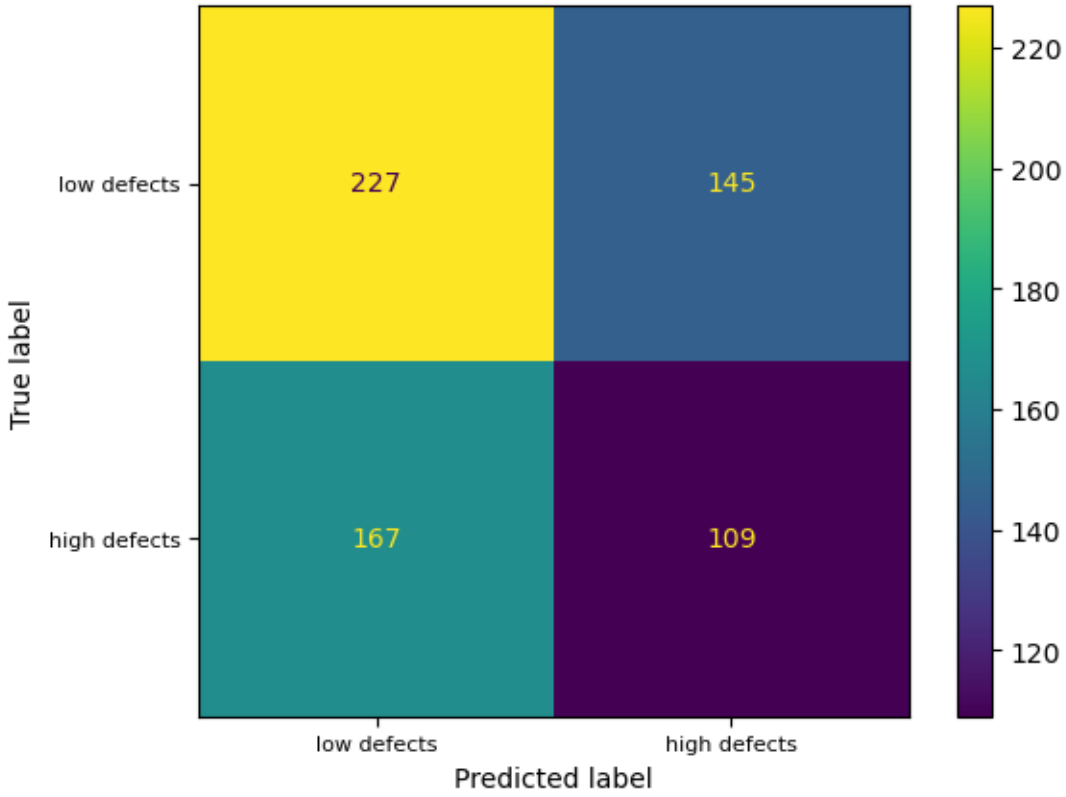


Figure 9. Confusion matrix for the gradient-boosted random forest. Here we see some improvement in the detection of high defects relative to earlier models.

Vd. Model Development with Support Vector Machines

For the support vector machine models (SVMs), we began with a simple model and gradually added refinements, as with prior models. For the first SVM the choice was to use an 'SVC' which means we allow only linear boundaries to separate the class of points in the dataset, using an rbf kernel (radial basis function which is very commonly used, and that tolerates non-linearity well).

The initial trial gave results that were somewhat better than a single decision tree, though not quite as good at finding high defects as the optimized logistic regression or boosted RF models. To optimize performance, a grid search over the parameters C and gamma was conducted, and the best results (C= 100 and gamma =.001) were used to create a new SVM. (In a nutshell, higher values of C like 100 push the model towards lower classification error at the expense of narrowing margin boundaries between classes, while a small gamma parameter allows the boundary to vary quite a bit, which may reflect noisy data).

The results are displayed in Figure 10, below. While the results are strong for avoiding misclassification of low-defect shipments, not many high-defect shipments were detected. The model may have some usefulness, then, as a 'conservative' model that tries to find only a fraction of bad shipments but otherwise doesn't create too many false alarms with low defect shipments. (That said, it is only correctly detecting high defects about 9% of the time!)

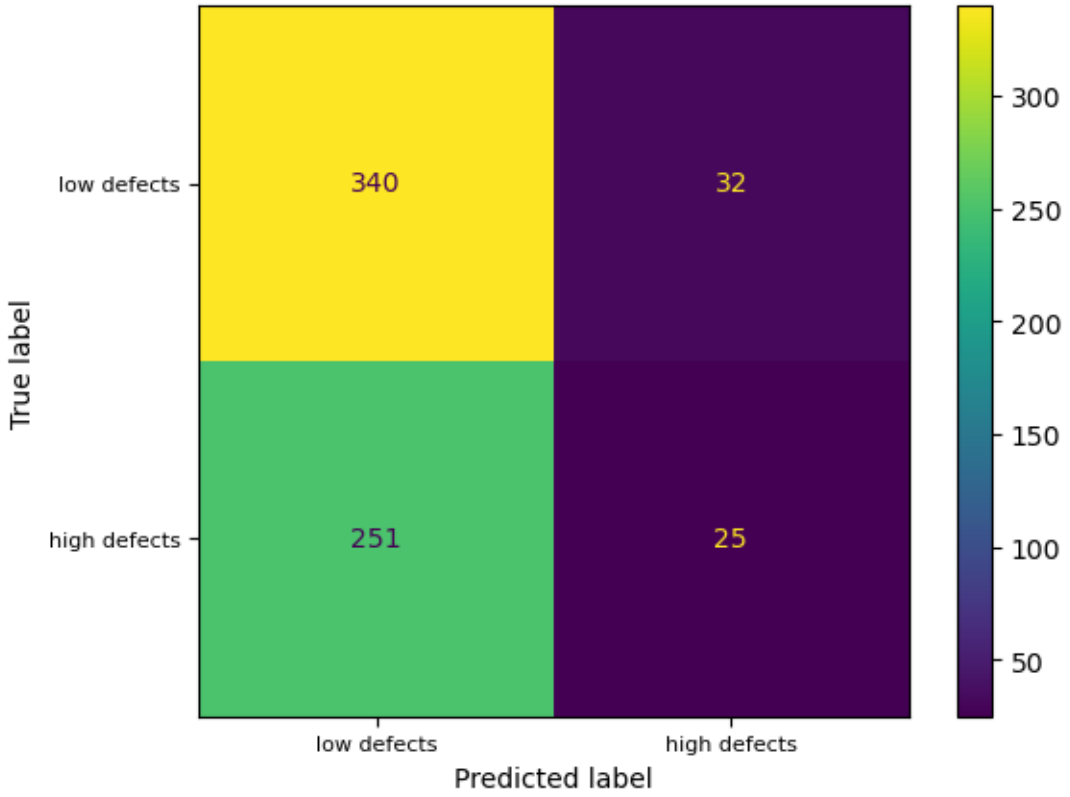


Figure 10. SVM confusion matrix using an optimized model. While low defects are mostly correct, high defects are only being found less than ten percent of the time.

VI. Conclusions and Recommendations.

Models for predicting manufacturing defects based on this dataset were only slightly better than random guessing. That said, depending on the situation, the models may still have some value. As remarked earlier, the models might be used to flag at least a fraction of the high defect shipments ahead of time. Perhaps such a model could be used to trigger immediate inspection upon arrival at an assembly facility, to reduce manufacturing delays or other production problems.

While prioritizing inspections would be one use of the model, another would be as an adjunct to existing models which rate suppliers but which do not already do so with any data pertaining to quality (relating to low defect parts). The latter approach is akin to extending credit to a consumer: a credit score is generated, but is only one piece of a larger system of originations which may take into account several other factors. In this way our model would serve as the input to another, more comprehensive model for rating suppliers.

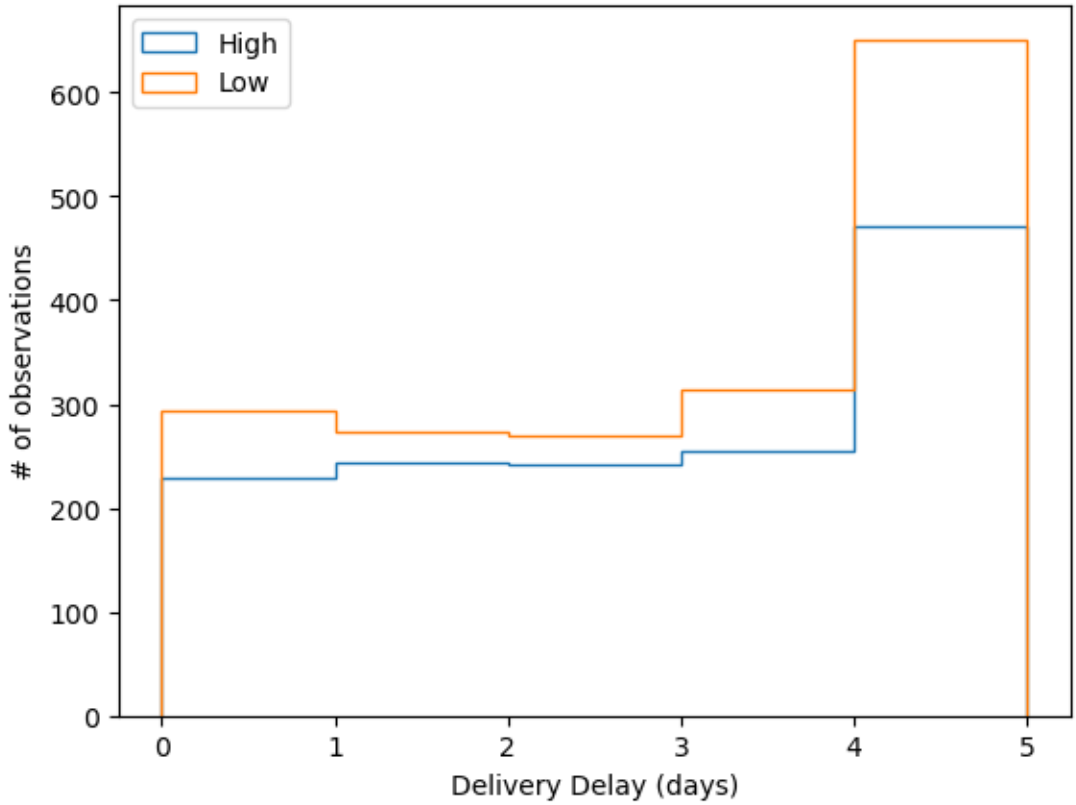
Some further improvements to the model may be possible. One might consider flagging only those orders with defect rates of 4-5 per thousand; it may be that this groups stands out more for classification, even if it makes the dataset less balanced for analysis.

Aside from changing the target as mentioned above to focus on identifying the worst of the high-defect shipments, it might be helpful to introduce engineered variables. The idea would be to assess whether combining some variables adds insight (e.g., taking ratios such as delay/production volume or other factors that might scale with production level). Finally, some other variations on SVMs (different hyperplanes or kernels) might be worth experimenting with. A separate file is submitted with final parameters for each tuned model.

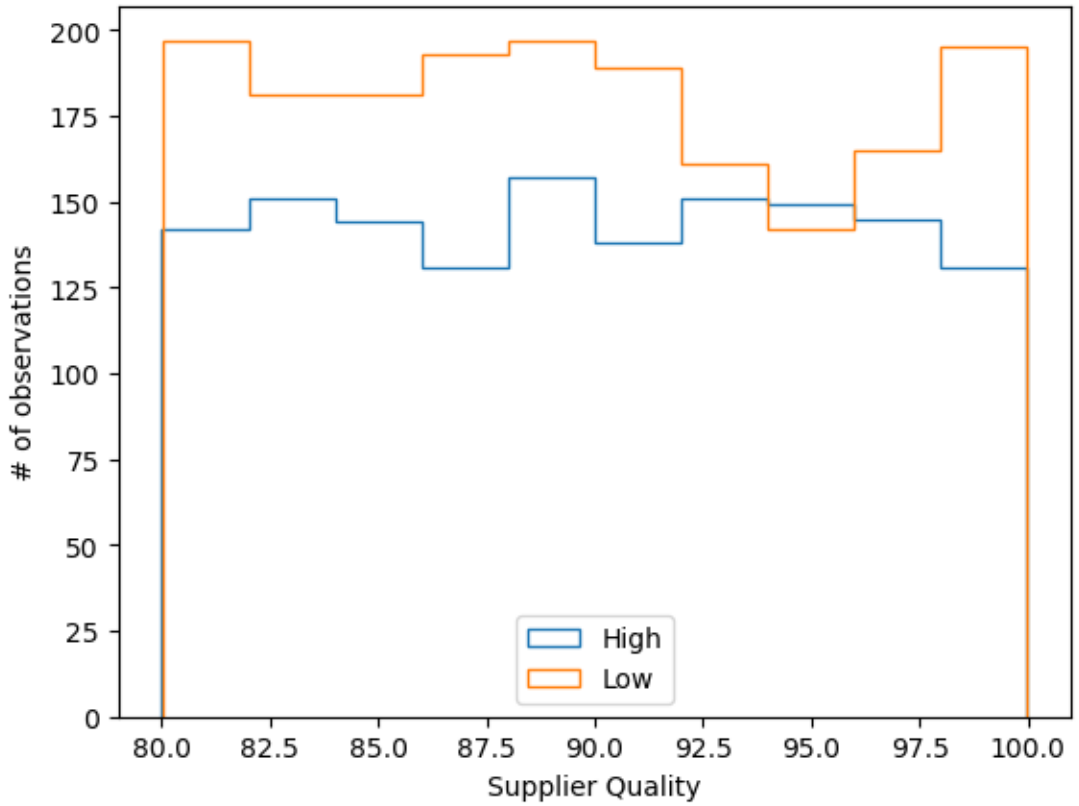
It was noted throughout that very little signal seemed to appear among the noise in the data variables, and that only a few features could be said to have some rationale relating to the rate and presence of defects (e.g., maintenance hours or delivery delays). Under the circumstances, success predicting manufacturing defects from this data was rather limited, but the modeling exercise may point the way toward finding other indirect measurements of quality which provide a better signal for future models.

Appendix

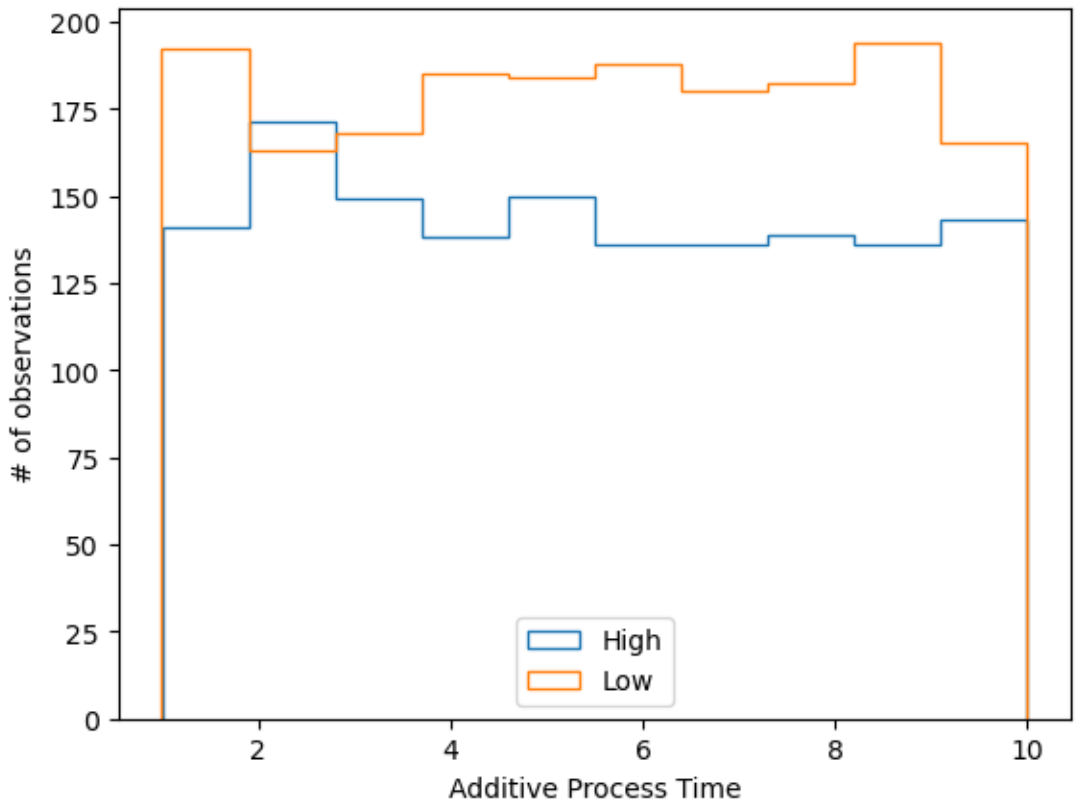
In this Appendix, we present selected graphs of supporting analyses. (This is not meant to be exhaustive; the complete set are presented in the Jupyter notebook).



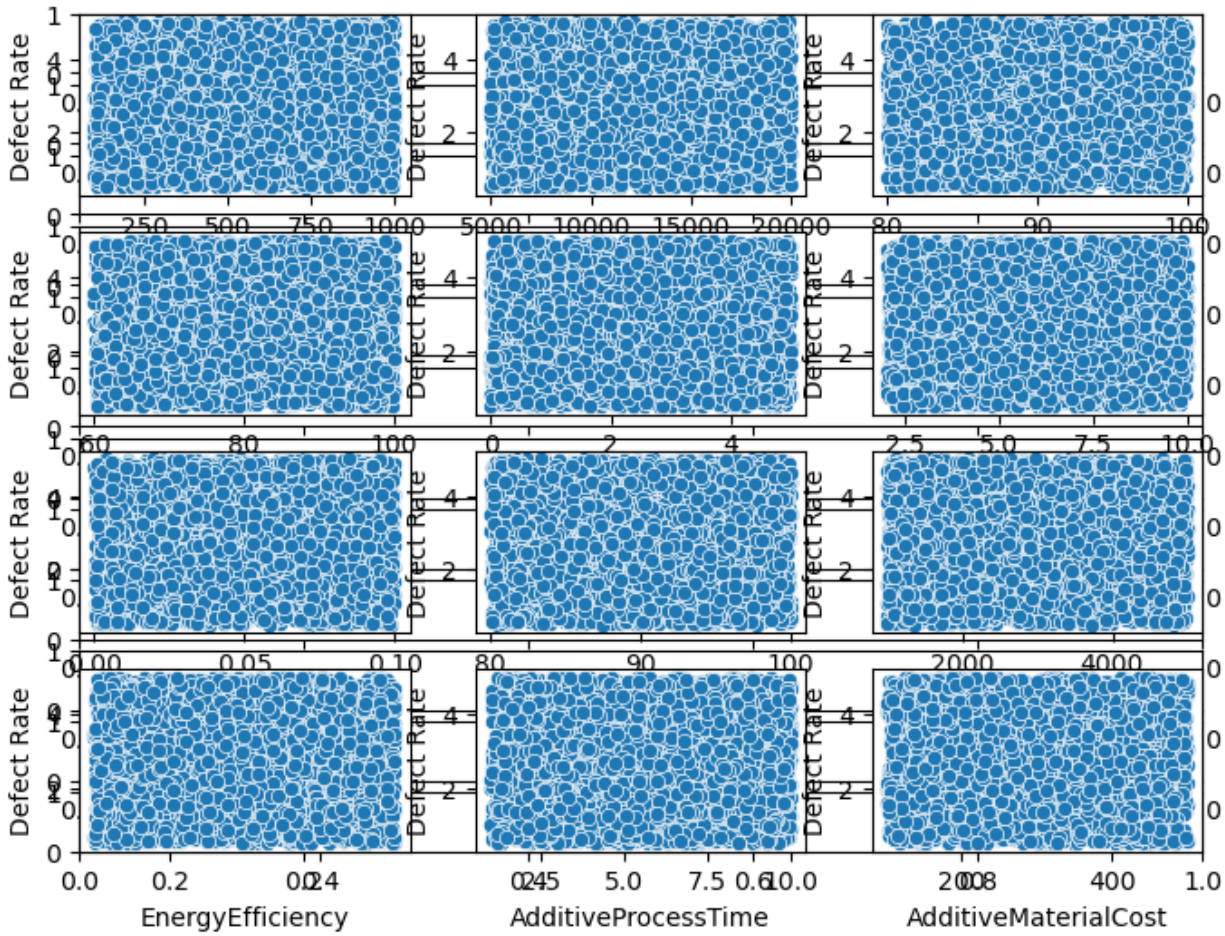
Appendix Figure 1. About twice as many shipments are delayed 5 or more days rather than fewer; the proportion of high/low may be different for long delays than not (where the proportion doesn't seem to change).



Appendix Figure 2. Ultimately Supplier quality rating was not included; the trend above included a strange dip at 95% for low-defect shipments, an observation for which a common-sense explanation is lacking, other than to assert this was noise.



Appendix Figure 3. High and low defect trends flatten out after 4 hours of additive time, but it is unclear why. Nevertheless, distinguishing trends like these were useful for the model.



Appendix Figure 4. For lack of space some captions are obscured, but the pattern is clear: all 12 scatter plots are very noisy! The weak relationship between measured defect rate and other continuous variables led to the decision to convert defect rate to a categorical variable (high vs low). The scatterplots from top left display: Production volume, Production cost, Supplier quality, Quality score, Downtime percentage, Inventory turnover, Stockout rate, Worker productivity, and Energy consumption (all against Defect Rate).