ISYE 6748 Applied Analytics Practicum Final Paper

Parker Jamison, Barbara Remmers, Xiaolu Su
pjamison6@gatech.edu, bremmers3@gatech.edu, xsu73@gatech.edu

Abstract—Several models that predicted a test result for linerboard rolls from a year of their production data are presented. Since the production process is intricate and the production data is noisy, distinguishing signal from noise is difficult. A multiple linear regression model, with two engineered features designed to reduce noise, outperformed a naïve model (that uses prior actual values for predictions.) Performance improved no further when grades of linerboard are modeled separately with any modeling technique. A tree-based model, LightGBM, exhibited superior performance. Although most Tree-based models struggled to predict, only they could supply valuable business information regarding which parts of the production process can influence the test results.

1 INTRODUCTION

Paper mills employ complex automated processes to manufacture and evaluate their products. Digital sensors collect vast quantities of real-time data.[1] The data has the potential to improve the efficiency and quality of production. The project utilizes one year of linerboard manufacturing data from the International Paper company to develop models that predict a quality test outcome.

1.1 Problem Description

Linerboard forms the outside layers of corrugated cardboard and impacts the strength and durability of the boxes made from it. Linerboard has standard industry grades and standard industry tests that assess important functional qualities. The test studied in this project was the STFI Short Span Compressive Test, which assessed compression strength. The test is also known as the STFI test (after the Swedish Test Fibre Institute, one of its developers).[8] International Paper uses an Autoline automated testing system, made by ABB, to perform the STFI test. The project built and evaluated several models that predicted the outcome of the STFI test from historical data.

Effective models can improve production efficiency by predicting the test outcome before the actual test results arrive. The predictions enable production adjustments without investing time and materials for the actual test results. Earlier adjustments can result in fewer rolls of low-quality linerboard being produced.

1.2 Business Assumptions

The ability to predict the STFI test results as early as possible for each linerboard roll can be valuable to International Paper. Accurate predictions will enable adjustments to production line settings that can improve the compression strength of the linerboard rolls and their STFI test results. Also, the company will save time by identifying defective rolls, which are destined for re-pulping, earlier in the process. Early identification enables an early start for making the subsequent roll and thus increases productivity. An accurate predictive model can not only reduce manufacturing costs but also improve the assessment of product quality. This lowers risks tied to the delivery of the intended product. Additionally, knowing which features the model uses to predict the STFI results is valuable. These features, together with domain expertise, can provide focus areas for the entire production line, and aid in the development of key performance indicators (KPIs) to ensure the manufacturing process is performing within tolerances.

1.3 Influential Research

The work by Bergmeir and Benítez (2012) in "On the use of cross-validation for time series predictor evaluation" proved to be very useful in describing various evaluation techniques of cross-validating modeling results using time series data. This paper provided the basis for the consideration of rolling window cross-validation as well as block cross-validation. The research also leads down the path of utilizing lagged predictor variables due to the autoregressive nature of the data.

2 DATA EXPLORATION & ENGINEERING

The data describes linerboard manufacturing at an International Paper facility in Rome, Georgia, from September 20, 2021, through September 19, 2022. The data set consists of 14,254 observations of 205 variables. Each observation pertains to one roll of linerboard as the roll is manufactured. The variables include six categorical variables, 198 numerical variables, and a datetime variable that serves as an index. The target variable, “PM1 Reel STFI CD Corrected Autoline Average | 93”, quantifies compression strength. International Paper performs the test on every other roll. The data has missing values for 54% of the target variable. Three variables are reported at the same time as the target variable and are therefore unavailable for predicting the current target value. All other variables are available to evaluate as predictors.

2.1 Exploratory Data Analysis

Figure 1 is a plot of the target variable in chronological order. It reveals several striking characteristics of the data and the underlying process that produces it.

The rolls, called grade runs, are processed in batches of a single grade, which are identifiable by color in Figure 1. Most grades are defined by their basis weight, which is the two-digit number seen in grade names. The actual weights of rolls are typically slightly less than the basis weight. The bulk (90%) of the rolls in the dataset is comprised of four grades as shown in Figure 1: HP56# (45% of rolls, lavender), 69# (22%, pink), 42# (17%, blue), and HP45# (6%, orange).

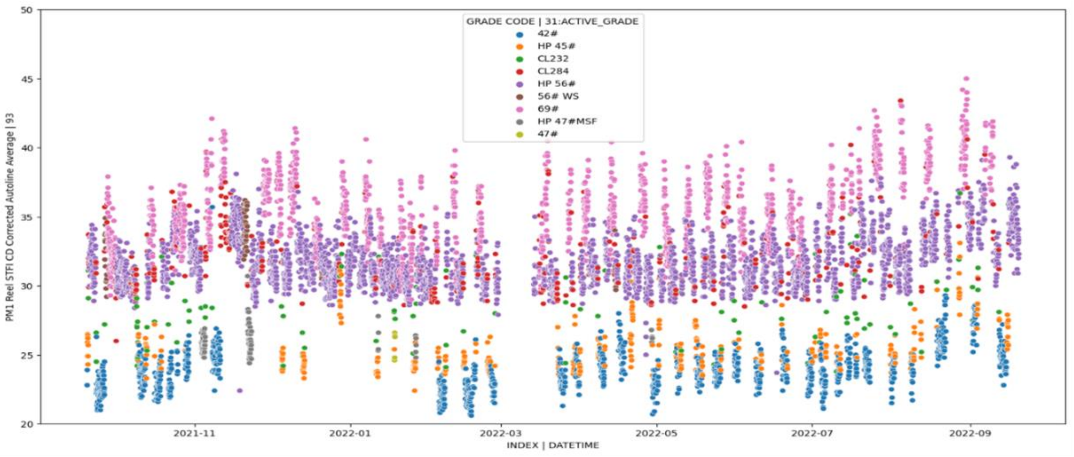

Figure 1—STFI data values in chronological order

The grades are clearly associated with the target variable but not its sole determinant. The general level (e.g., mean) of the target value changes from grade run to grade run of the same grade, as seen in Figure 1 where runs of the same color rise and fall from run to run. There is also substantial variation in target values within a single grade run. Some grade runs, viewed on a smaller scale, exhibit a trend line or curve in addition to noisy variation.

Two grades, CL232 and CL284 (green and red in Figure 1), are transitional grades and represent 2% and 6% of the sample, respectively. The transitional grade rolls have highly varying target values that start near the level of the prior grade and end near the level of the subsequent grade. One, CL232, appears immediately before and after grade runs with basis weights below 50. The other, CL284, appears in two contexts. The first is immediately before CL232 during transitions from grade runs with basis weights below 50 to grade runs with basis weights above 50. The second context in which CL284 appears is during the transition between the two highest basis weight grades, 69# and HP 56#, where it is the sole transitional grade. The three remaining grades are *de minimis* (56# WS, 47#, and HP 47#MSF) together comprising 2% of the rolls.

The gap in the middle of the plot shows the absence of data for 13 consecutive days in March 2022, when production stopped for annual maintenance. Upon restarting the variation in the target variable was higher than usual through the first grade run of 69# rolls. The brief unstable period was expected upon restarting. The data also has periodic single-day or shorter pauses in production where the data is not noticeably unusual both before and after the pause.

2.2 Data Challenges

The large number of features, 205, precluded exploratory data analysis (EDA) of the sort that considers each feature individually and in relation to the target variable. Instead, EDA began with the plot depicted in Figure 1 and aimed to infer notable aspects of the manufacturing process that produced the data. Elements of EDA continued with the assessment of outliers and missing values, are described in the data cleaning and feature engineering sections.

A significant issue with the target variable is that 54% of its values are missing. This mainly results from the test being performed on only one roll of each consecutive pair of rolls after they are manufactured. After consultation with a subject matter expert, the observations lacking a value for the target variable were eliminated and modeling proceeded with 46% of the original data. The resulting dataset had an overall missing value percentage of approximately 10% and these values were accounted for during modeling using a variety of techniques or otherwise ignored if the modeling algorithms allowed for missing values.

2.3 Data Cleaning

Categorical Variables

5 of the 6 categorical variables included in the original dataset were discarded. 2 of them were redundant and the other 3 were (virtually) single-valued. See Appendix 2 for names and justification of eliminated variables.

Numerical Variables

20 of the original 198 numerical variables were discarded, 4 of which were single valued while the other 16 had more than 10,000 missing values. See Appendix 2 for names and justification of eliminated variables.

Date-Time Variable

The dataset was sorted based on the chronological order of the date-time index variable.

Missing Values

Ten missing grade codes were imputed, using judgment based on their weights and similar patterns in nearby data. This decision balanced the risk of introducing errors via mislabeling with the drawback of introducing gaps in the time series.

Roughly half of the observations were eliminated since the target value was missing because only every other roll was tested. Additionally, 86 observations following the planned annual two-week production pause were eliminated to remove the unusually varying data that was attributable to restarting.

2.4 Feature Engineering

Lagged Autoline Test Variables

Three variables reflecting additional Autoline test results conducted at the same time as the STFI test are "PM1 Reel Mullen Average Autoline Average | 57", "PM1 Reel Gurley Porosity Autoline Average | 39", and "PM1 Reel TSI MD/CD Autoline Average | 129". They were lagged at 1 lag so that they could serve as predictors. The lagging was necessary because the only Autoline test results that would be available to predict for the current observations were the prior results.

Target Variable

Before eliminating observations with missing target values, a feature, 'minutes_since_previous' was created to preserve the manufacturing times of the tested rolls.

A lagged target variable, 'lagged_y' was created. Although the target variable itself was not altered, the lagged target variable was cleaned to eliminate outliers that would particularly hamper MLR models. The problem of identifying bad values of STFI was nontrivial since typical values vary by grade and time. With the goal of a data cleaning method that would be possible in real-time (i.e., if the model were implemented in an online fashion), a judgment-based rule was employed. All lagged STFI values for grade 60# that were outside the range of [31, 41] and the values for grade HP 56# that were outside the range of [29, 36] were forward filled with the most recent prior value. The arbitrary nature of the rule was balanced by the ease of implementation. Improving or further evaluating the rule is left for future work. The cleaning rule adjusted 163 observations.

Weight Variables

Weight variable cleaning employs a similar rule: weights more than 5% different from a roll's basis weight are replaced by missing values that are then eliminated by forward filling. Data checks suggest that the unusual values seem more likely to result from errors of some type than from the manufacturing process suddenly changing and then resuming typical behavior. For instance, there are weight values of zero in observations with typical compression test values, which is physically impossible. The weight cleaning rule adjusted 78 observations. A lagged weight variable, 'lagged_wt' is created with one lag.

Engineered Features

The engineered feature, y.eng, aims to estimate the current observation's target value with only prior STFI values. The feature depends on where in the grade run the observation is located. The feature for the observation associated with the first roll of a grade run is set to the lagged STFI value. Although this value is from the prior grade run, it is often close to the values for the current grade, particularly if it is from a transitional grade. The feature for the observations associated with the second through twelfth rolls of a grade run is set to the average of all prior STFI values in the same grade run. The feature for the observations associated with the thirteenth and later rolls of a grade run is set based on the preceding 12 STFI values. The ordinary least squares algorithm uses the 12 preceding values to estimate the current value. The parameter that determines whether the feature is an average or a linear estimate is set to 12 based on cross-validation. Higher parameter values improve the accuracy of the slope estimate; lower ones quickly capture changes in the trend of the target value. The data's noise, combined with higher-degree extrapolation issues, motivates this method. While linear, it allows for the influence of non-linear trends of the target variable within a grade run.

The engineered weight feature, w.eng, compares the weight of a roll to typical weights earlier in the same grade run. It is the current observation's weight minus the average of the weights of all the preceding rolls in the same grade run. For the first roll of a grade run, it is set to zero.

3 PROPOSED METHODOLOGIES

The modeling task was approached from two directions: bottom-up and top-down. The two approaches were synergistic.

The bottom-up approach starts with the naïve model, where the predicted value is the previous value, and then aims to improve upon it, in the setting of multiple linear regression (MLR). The simple setting allows for straightforward comparisons during feature engineering. Graphing residuals against unused variables can suggest additional features, as can the results of the top-down models. The relatively simple models provide a performance floor for more complex models.

The top-down approach starts with models that can consider all the available data from the start: lasso, Random Forest, XGBoost, and LightGBM. This enables the discovery of relationships with the target variable that may be hard to discern otherwise because they are highly non-linear, interactive with other variables, or otherwise complex. Along with the potential for high performance, these models can identify variables to consider adding to the simpler models, or that are candidates for feature engineering.

3.1 Linear Models

Linear models remain highly useful for modeling problems even when complex models, like the tree-based ones here, offer advantages when confronting high-dimensional data. With a known starting point, such as the Naïve model, model deficiencies can be studied and ameliorated, leading to an improved linear model. Transformations, engineered features, higher-order terms, and interaction terms all reduce the restrictions imposed by linearity. Cleaning the data limits the influence of outliers. The same restrictions that create challenges to capturing nuances of relationships between dependent and explanatory variables also serve to mitigate overfitting by limiting how much idiosyncratic noise is captured from the training data. Linear models are also able to extrapolate whereas predictions from tree models remain within the range of target variables in training data.

3.2 Tree-Based Models

Tree-based models are appealing because they can handle high dimensionality, non-linear relationships, and interacting variables. Single-tree models tend to overfit, which led to the development of more robust models that employ multiple trees. Since the relative performance of different models on different data and modeling tasks is not known in advance, three tree-based models were estimated and evaluated.

3.2.1 Random Forest

Random forest models employ a forest of trees where each uses a different random subset of data and where splitting at each node considers a random subset of features. The model predictions are averages of the predictions from the individual trees. The bagging and averaging ameliorates overfitting. Compared to other ensemble tree models, it is relatively easy to tune and fast to estimate. Results are sensitive to tuning parameters since all trees are affected by them. It is often outperformed by boosting models, but for any given problem and dataset it might exhibit superior performance.

3.2.2 LightGBM

The first proposed Gradient Boosted Decision Tree (GBDT) algorithm is LightGBM which uses leaf-wise (best-first) tree growth. LightGBM is able to maintain accuracy and speed in model training with large datasets owing to three aspects.

First, its high accuracy is bolstered by its leaf-wise tree growth. At each step, the decision tree grows by choosing the best subset among all subsets as the best node, and it keeps branching in this manner until the leaves or terminal nodes are pure unless the expansion threshold is met (Shi, H., 2007). Compared to depth-wise tree growth used by XGBoost, leaf-wise tree growth ensures each split will make the most contribution to minimizing the global loss, which can result in learning decision trees on massive data with low errors.

Second, other than its leaf-wise characteristics, LightGBM uses a Gradient-based One-Side Sampling (GOSS) algorithm to optimize the accuracy in splitting data instances at nodes (Guolin K. et al., 2017). This algorithm sorts all data instances with their corresponding gradients. Then it retains the large gradient samples while randomly selecting and assigning weights to the small gradient samples (Wang, R. et al., 2019). The purpose of the algorithm is to maintain the distribution of the original dataset as much as possible so that the model could better fit the data.

Furthermore, to efficiently train the models, LightGBM can accelerate the training process by using a technique called Exclusive Feature Bundling (EFB), because it can create a new predictor by bagging mutually exclusive features among sparse features. It does an excellent job of reducing dimensionality when dealing with high-dimensional data. However, LightGBM is subject to overfitting because the leaf-wise tree growth tends to find patterns in the training sets that may not exist in the test sets. Thus, it can be less robust than XGBoost. Due to the tree-boosting characteristics, the interpretation of the LightGBM model will be challenging.

Last, the modeling process of LightGBM will also use the final dataset from 2.3 Data Cleaning. The algorithm will be implemented for two schemes: all-grade and single-grade. The models will fit the whole dataset for the all-grade scheme while separately fitting different subsets filtered by the grade code for the single-grade scheme.

3.2.3 XGBoost

The next GBDT algorithm chosen is XGBoost. This method works with regression or classification and requires a certain amount of data preparation before getting started. XGBoost uses decision trees as its base model technique with each tree splitting the data into smaller groups based on the features provided. Many trees get built and boosted which adjusts the predictions of each decision tree based on each tree's performance. This is achieved by minimizing the loss function, Root Mean Squared Error, for this project. Each tree is then weighted, and their predictions are adjusted by the weight. The more accurate trees will receive a higher weight. The combination of these weighted predictions as an ensemble model is expected to result in a more accurate prediction. The XGBoost package allows for tuning of hyperparameters which when properly adjusted should increase model accuracy. A full overview of the hyperparameters that were tuned can be found in the XGBoost documentation found in the References section.[2]

3.3 Cross-Validation

Cross-validation in a time series context involves special considerations. The training folds must be defined as a collection of consecutive data points. The test data for a given fold must occur after the training data. Time series models typically contain lagged target variables as predictive features. Any use of backward-looking features involving lags of the target variable requires setting train/test splits with a gap between them. The gap must be wide enough to prevent any one target value from serving as a target value in the training data while also appearing as a feature (or embedded into an engineered feature) in the test data.

Two types of cross-validation, both appropriate for time series, appeared in this project. One was *rolling window* (also known as *walk forward*) and the other was blocked. Many variations of the two types of schemes exist. The basic version of each type is described below.

3.3.1 Rolling Window Cross-Validation

Rolling Window cross-validation is illustrated in Figure 2. In this type of scheme, training sets always begin with the earliest observation in the data set. The training set grows by a fixed number of observations with each fold. The test set is a constant size and occurs immediately after the training set and any gap. The scheme benefits from the large size of the training set, which allows for more learning during training. The cost of this benefit is that the scheme uses and reuses the data from the beginning of the data set, and therefore overemphasizes its influence on the model, compared to that of observations that occur later in the dataset. For homogenous datasets and processes, the issue is minimal. Less homogenous data is more affected by the overemphasis, but it also may require more training data to fit the model well.

3.3.2 Blocked Cross-Validation

Blocked cross-validation is illustrated in Figure 3. Each fold uses a block of consecutive observations that were further divided into a training set and a test set for the fold. Observations in the dataset appear in only a single fold as training data, and possibly in only one other as testing data. The scheme is suited to models that can be well fit using only a fraction of the available data. The scheme benefits from the occurrence of training and test sets that are evenly spread throughout the dataset. No particular stretch of data is implicitly over or under-emphasized when averaging error measures over folds.

3.3.3 Preventing Data Leakage Between Training and Testing Data

If the features involving lagged target values only use target values that are single-lagged, then the features in the test set only reach back in time one observation. In that case, a gap of a single observation between training and test data will suffice to keep the same values from affecting both training and testing. This is the situation of the two boosted models.

Using a feature that uses lags of the target variable that reach back to the start of a grade run requires a gap that extends back through one period before the beginning of the current grade run. This complicates the gap construction. Setting testing data boundaries at the start of a grade run and maintaining a gap of one unused observation between training and testing data eliminates the least data while also preventing lagged values from affecting both the training and testing sets.

Requiring grade runs and train/test splits to coincide, however, is only practical for schemes using a small number of folds. Then the boundary adjustments do not lead to testing sets with widely varying proportions of data. For this dataset, boundary adjustments are workable with six or fewer folds but become untenable as the number of folds increases. Schemes employing more than about 10 folds must remove any features with lagged target variables that reach back to the start of a grade run. The number of folds where the boundary adjustment becomes infeasible depends on the particular cross-validation scheme. For the schemes used in this project, boundary adjustments are either clearly feasible (linear models, random forest) or infeasible (boosted trees).

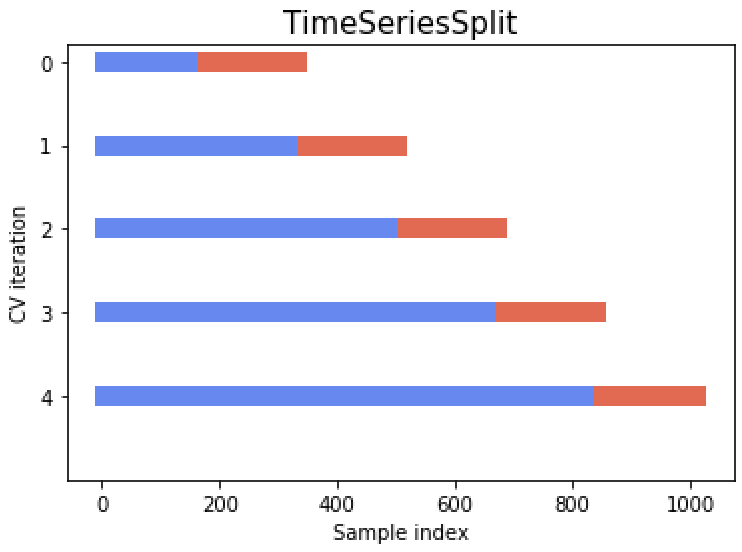

Figure 2— Rolling Window Cross-Validation Visual[4]

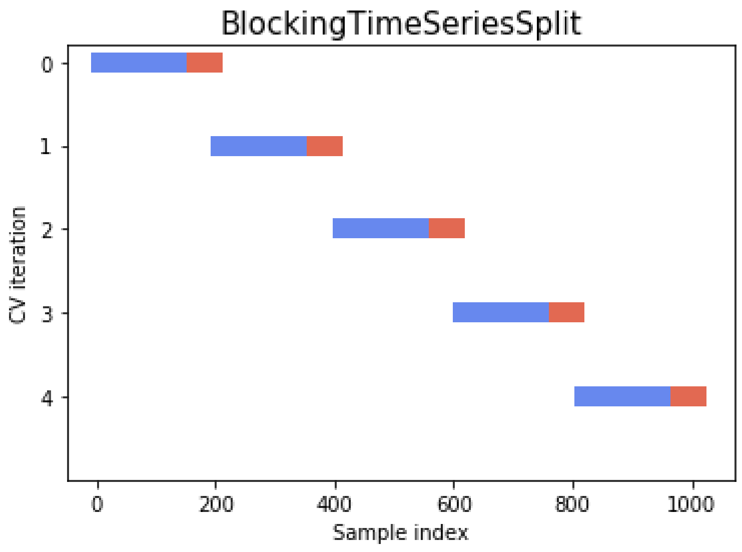

Figure 3— Blocking Window Cross-Validation Visual[4]

4 ANALYSIS & RESULTS

4.1 All Grade Linear Model Analysis and Results

4.1.1 Cross-Validation Analysis

Scheme

The cross-validation scheme is a variation on rolling window, having multiple non-overlapping testing sets for each fold. The multiple test sets allow for one fit of the model to be compared to different stretches of data. They also allow comparison of several model fits on the same stretch of testing data. In a setting with complex data, they help distinguish between model effects and data effects.

First, the non-holdout data is sliced into five units of contiguous observations of equal length. The units can then be used, in adjacent groups, as training and test data. The hold out data, although unused during model selection, can be defined as a sixth fold after the final model is decided for the purpose of generating illuminating comparisons along with the hold out error values.

Defining the units starts with dividing the non-hold out data into the desired number of (approximately) equal-sized initial units, and then proceeds to move the boundaries between units earlier in time. The first observation of a non-initial unit must coincide with the advent of a grade run. The last observation is two observations before the first grade run of the next unit. The eliminated observation has a target value that affects a feature in the test set. The elimination prevents the value from appearing in the training set.

This results in a collection of consecutive units that can be used singly or in conjunction with other units as either training or test sets. When using multiple units of data as a training or a test set, the gaps between single units are no longer needed.

Cross-validation begins by training the model on the first unit and testing it on each of the subsequent four units. It continues with training sets consisting of the first two blocks, then the first three blocks, etc. The trained models are then tested on each of the remaining future blocks of data.

A gap in the production of a common grade of linerboard precludes the use of blocked cross-validation as a primary scheme. Blocked schemes would have allowed multiple single-block training sets, and additional comparisons analogous to those afforded by the multiple testing sets. There is a significant stretch of time, stretching across an entire block for six folds, when the third most common linerboard grade, 42#, is not produced. The gap impairs models trained on that data when predicting 42# observations in subsequent test sets, since the regression coefficients are uninfluenced by the presence of any such observations. Since cross-validation aims to gauge the performance of a model that will be trained on all grades, the blocked schemes introduce an irrelevant hindrance to assessing the eventual performance accurately.

Analysis

The Naïve model (See Appendix 3 for details) supplies a base for the analysis. The engineered feature, y.eng, and its parameter of 12 were developed, followed by w.eng. The fits of the intermediate and final models appear in section 4.1.2. Since y.eng aims to estimate the target based on the target history, the variable w.eng aims to incorporate information that is not contained in that history. So w.eng is a measure of how much the current weight differs from the history of weights. The design is based on the idea that the information in the history of weights is already incorporated into the target history.

The models are estimated without intercepts. Intercepts overfit in this data. They allow a closer fit in the training data through the use of an extra variable and then hurt the fit in the testing data. Because the average target value in test data will be heavily influenced by the mix of grade codes in the test data, and the mix of grade codes varies, a constant term, by its very nature, mutes the adaptation to a novel mix of grades.

A variety of candidates for predictive features were studied for inclusion. The candidates were of two types. One type involved the existing features, and included higher order terms, transformations, and interactions with both the other engineered feature and as yet unused variables from the data. None merited inclusion. The second type involved variables recognized as important by the tree-based models. Several of these variables did have significant MLR coefficients, however, the model fit and performance were unchanged. The included variables had the effect of reducing the coefficient of y.eng, but not w.emg. This suggests that the candidate variable contributes only information that is also embedded in y.eng. The low correlation between y.eng and w.eng of 0.02 also indicates that w.eng is contributing new information to the model.

4.1.2 Results

Table 1 — Linear Model results.

Model	Predictors	Coefficients	RMSE Train	RMSE Test	MASE Train
Naïve	Lagged_y	1.00	1.57	1.57	1.00
OLS	y.eng	1.00	1.49	1.52	0.92
MLR	y.eng, w.eng	1.00, 0.33	1.46	1.49	0.91

4.1.3 Single Grade Modeling

Single-grade modeling has the potential to improve performance via coefficients tailored to each grade but has to estimate the coefficients using a smaller amount of data than all grade modeling. In this case the only parameter likely to change is the coefficient on w.eng, which is less influential on performance than y.eng.

The three *de minimis* grades are ignored in this analysis. They do not appear in the hold out data.

The single-grade modeling could not use the fivefold cross-validation used for the all-grade model due to both data size and alignment concerns. Instead, the non-hold-out dataset was split in half, with the first half used for training and the second half for testing. The same was done for the all-grade model to supply comparable test errors. The errors appear in Table 2.

Table 2 — All grade and single-grade modeling results

	All Grade Model Scores				Single Grade Model Scores			
Grade Code	**RMSE Train**	**RMSE Test**	**MASE Train**	**MASE Test**	**RMSE train**	**RMSE Test**	**MASE train**	**MASE test**
69#	1.53	1.67	0.98	0.94	1.55	1.73	1.00	1.00
HP 56#	1.26	1.33	0.90	0.92	1.38	1.41	1.00	1.00
42#	1.32	0.99	0.84	0.93	1.34	1.14	0.98	1.02
HP 45#	1.15	1.22	0.87	0.99	1.12	1.26	1.00	1.01
CL232	3.06	3.73	0.95	0.91	2.99	3.73	0.96	0.93
CL284	2.07	2.27	0.94	0.89	2.16	2.25	0.97	0.96

The individual grade models do not model their own grade better than the all-grade model, and no better than the Naïve model for the non-transitional grades.

4.2 Tree-Based Model Results

4.2.1 Random Forest

The Random Forest model was uncharacteristically difficult to tune using the same cross-validation scheme as the linear model. Testing RMSEs and MASEs dropped precipitously as more training data was added although training errors held steady. Therefore, tuning relied on training with the first four-fifths of non-hold out data and testing on the last fifth. (Also, single-grade modeling was not attempted.) The final model appeared reasonably fitted as seen from the values reported in Section 4.3, Table 7. The test RMSE of 1.45 was higher than the training RMSE of 1.35, but the test MASE of 0.88 was lower than the training MASE of 0.93. When presented with the hold out data, the model predicted poorly, with RMSE of 2.02 and MASE of 1.16.

The model not only performed worse than the Naïve model but also gave no insight, beyond that provided by the MLR model, into which parts of the production process merit particular scrutiny because the influential features were all engineered ones based on either lagged targets or weights. Removing such variables from the data resulted in a model that performed even worse but did identify which production variables the algorithm determined to be most important for predicting the target variable. The top 10 most influential sensor data variables and their relative influence appear in Figure 4. The training RMSE rose to 1.47 from 1.35 when the engineered features were removed, which suggests that the digitally produced production data, despite its many variables, does not capture enough important aspects of the complex production process to accurately predict the compressive strength of the product.

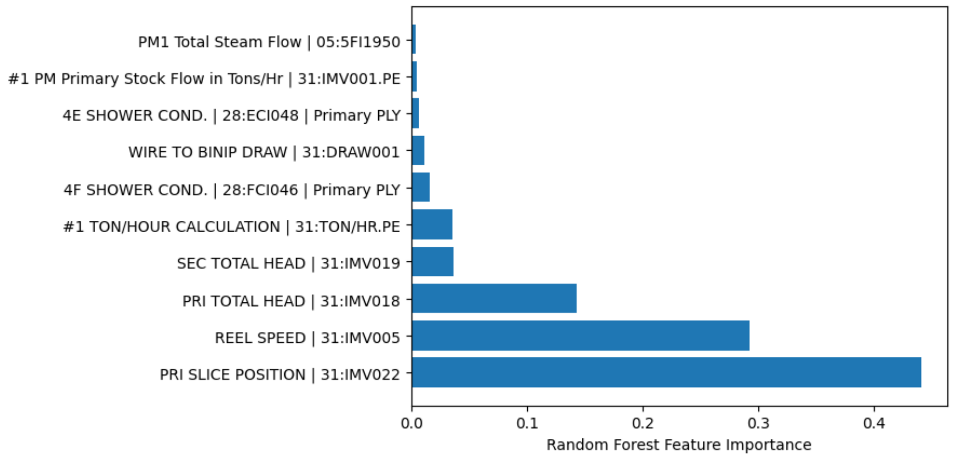

Figure 4—Influential sensor features from the Random Forest model

4.2.2 LightGBM

To continue forecasting the STFI target variable using Gradient Boosting Decision Trees (GBDT) of the tree-based models, the package first applied to the model was LightGBM for all grades of linerboard. The model was tuned with four parameters: the number of folds (ranging from 5 to 50), the maximum number of leaves (ranging from 3 to 11) on each tree, the minimum data on each leaf (ranging from 3 to 11), and the index of an individual fold.

Model Selection Rule 1: Fold Index = -1

After training 98,415 different LightGBM models, it was tempting to simply sort the results by the Root Mean Squared Error (RMSE) on the test set (RMSE_test) of all models, because RMSEs on both training and test sets were smaller than 1, and the Mean Absolute Scaled Error (MASE) on the test sets was around 0.57 – 0.62. Nevertheless, this simple sorting was a problematic approach. For the top 20 models selected, the RMSE_train was always larger than the RMSE_test, so the test sets seemed to have made it too easy for the models to predict. Moreover, these models all fitted on the 24th fold of the 44-fold rolling window cross-validation. Not only was the train-test set too specific to justify the proper fitting of models, but the 24th fold of the cross-validation used only 55.5% of the available training observations. Any of these top-selected models would be too weak to forecast the target value in the hold out set regardless of their tree structures.

The previous model selection failure encouraged choosing models from the final fold of the rolling window cross-validation because training the most amount of data and testing the models on the most recent data before the timestamp of the hold out set could always be beneficial.

Model Selection Rule 2: RMSE_train, RMSE_test < 1.5

As for the issue of having the RMSE_train > RMSE_test, even though it would lose a lot of important information to force the model candidates to have RMSE_train < RMSE_test, it gave a warning that some inferior models could affect the model selection. Motivated by the analyses from 4.1 Linear Model Results, since the practice of LightGBM was intended to find improvements in results, the second filter applied to the model selection was limiting both the RMSE_train and RMSE_test to be lower than 1.5.

Model Selection Rule 3: Optimal K-Fold by Minimum Average and Median RMSE_test

Once the model outputs were filtered by the fundamental rules, the model needed to find the optimal train-test split ratio or the number of folds to continue the model selection. The following graph of the average and median RMSE_test by fold (Figure 5) showed that the average and median of RMSE_test generally had an agreement on model performance by fold and suggested n_fold = 26 to be the best fold number given the minimum average RMSE_test of 1.24 and minimum median of 1.22. Looking at both the average and median could help avoid choosing the n_fold affected by outliers. After the split, LightGBM would be training on 5,524 data instances and testing on the 65 ones.

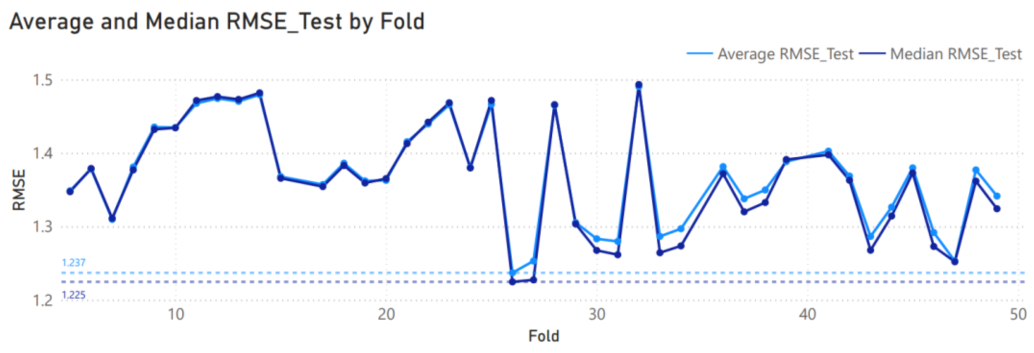

Figure 5—Average and median of RSME on test sets by the number of folds in cross-validation.

It was also informed in Figure 5 that LightGBM seemed to need at least n_fold = 26 for the models to have a better understanding of the training sets in order to interpret the patterns in the test sets. But as n_fold further increased, the RMSE_test fluctuated severely until it was relatively stabilized when n_fold = 34. The fluctuation could result from the dramatic change of variance in training and test sets by different splitting locations, whereas the comparatively stabilized RMSE_test starting from n_fold = 34 might have worse overfitting issues because the test sets would get only 50 data instances to test the models. Therefore, n_fold = 26 was preferred to keep the train-test sets at a more reasonable size.

Model Selection Rule 4: Select the Minimum Data in Leaf by Fold, sorted by the lowest RMSE_test

When n_fold = 26, LightGBM worked significantly better when the leaf size or minimum data in leaf Min_Data = 3 compared to other leaf sizes given that it had the lowest average RMSE_test of 1.22 (Figure 6). The low average of RMSE_test acknowledged the quality of the performance, but it could not be guaranteed until the variance of RMSE_test was considered. Since the variance was shown to be generally aligned with RMSE_test by Min_Data in the stacked area chart, when Min_Data = 3, the LightGBM models would produce the best results with high certainty. Note that the median of RMSE_test was no longer used because as the outputs got sized down by filters, it started to show very similar behaviors as the average of RMSE_test, which made it a more helpful choice to refer to the variance.

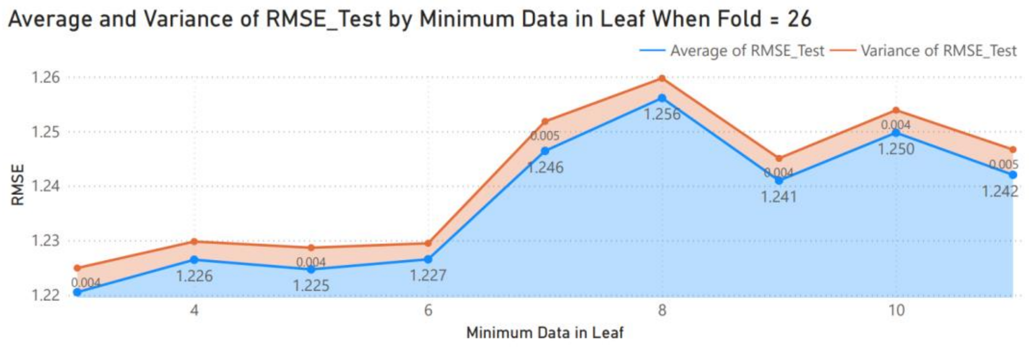

Figure 6—Average and variance on test sets by the minimum data in leaf when the number of folds equals 26.

Model Selection Rule 5: Select the Maximum Number of Leaves by the Minimum Data in Leaf

The last step was to find the optimal maximum number of leaves in a decision tree. Similarly, the optimal number was decided by the minimum average RMSE_test and the variance of it. Figure 7 suggested that when the maximum number of leaves = 10, the models generally performed better on all different rolling window cross-validation. Although there was not much difference among the average RMSE_test when Leaf = 9, 10, or 11, and the variance also indicated some uncertainty among the three, it did not affect selecting Leaf = 10 to be the optimal maximum number of leaves in each decision tree based on its lowest average of RMSE_test.

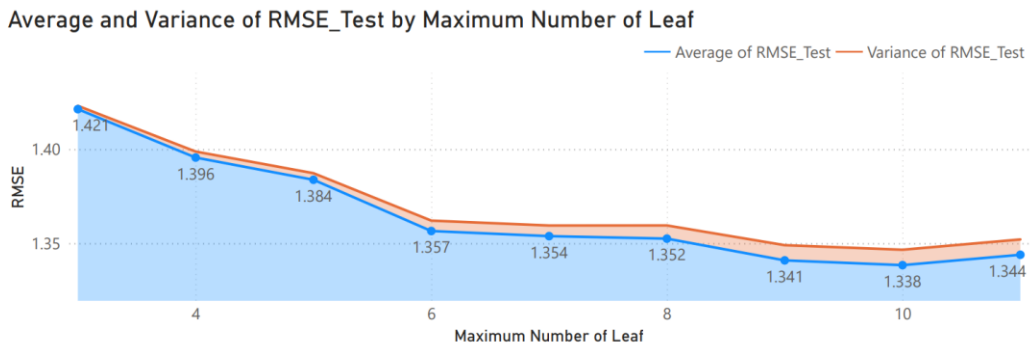

Figure 7—Average and variance of the RMSE on test sets by the maximum number of leaves in each decision tree.

Note that the number of leaves was selected regardless of the number of folds for two reasons. For one thing, the number of leaves was more closely related to the minimum data allowed in each leaf while less concerned with the sizes of the training and test sets. For another thing, it would be more helpful to use the global knowledge gained from the fairly good models (Fold_id = -1, RMSE_train, RMSE_test < 1.5) trained in all of the K-fold cross-validations instead of finding the local optimal.

Clarification on Rules 4 and 5

Rules 4 and 5 suggested that the optimal minimum data in a leaf should be determined before finding the maximum number of leaves in each decision tree. Although it might seem to be an alternative to reverse the order of these rules (deciding the optimal maximum number of leaves before choosing the number of minimum data in a leaf), it potentially violated the tree-pruning process. In a decision tree, the minimum data in a leaf was normally set as a threshold to control the growth of the tree. If a terminal node had less data than the minimum threshold, this node or leaf should be trimmed. Additionally, using the reverse order could result in a higher uncertainty at the end of the model selection as shown in Figures 8 and 9 in Appendix 1.

Summary of Model Selection and Results

In summary, the final LightGBM model was selected by the following criteria in order: models should be tested on the last fold of data in the rolling window cross-validation; RMSE_train and RMSE_test should be constrained within a reasonable range; the optimal K-Fold should be determined by the minimum average and median RMSE_test; the minimum data in a leaf should be selected by fold and sorted by the lowest RMSE_test; and finally, the maximum number of leaves should be determined by the minimum data in a leaf regardless of the number of folds.

Among all the models trained, the final LightGBM model (Table 3) had a reasonable 0.203 increase in RMSE from training to test, while the Mean Absolute Scaled Errors (MASEs) on the training and test sets were also within a good range and similarly small. Even though both the RMSE_Holdout and MASE_Holdout were significantly higher than the ones on the test set, MASE_Holdout = 0.93 reflected that the final LightGBM model would produce better forecasts compared to the Naïve model, which used the lagged target values as the predicted values. Besides, it only took 0.246 seconds to run such a complex model.

Table 3 — Parameters and outputs of the final LightGBM model.

K-Fold	Min. Data	Max. Leaf	Fold id	RMSE Train	RMSE Test	MASE Train	MASE Test	RMSE Hold Out	MASE Hold Out	Time (sec)
26	*3*	*10*	*29*	*0.95*	*1.16*	*0.64*	*0.63*	*1.56*	*0.95*	*0.25*

The top 15 features selected by the final LightGBM model rated highly of three engineered features: the previous target value (lagged_y), the previous Reel Autoline Average | 129, and the cleaned weights (cleaned_wt, refer to 2.3 Data Cleaning) (Table 4). In fact, clean_wt was constantly used as the first node among the 100 decision trees generated by the model and lagged_y was no doubt dominating most levels of the first decision tree (Figure 8). Thus, the rankings of the engineered features suggested that the final LightGBM model could notice the contribution made by the cleaned weights to the training accuracy and it found the previous Reel Autoline Average | 129 to be somehow related to the target variable. And the model would regularly review the previous target value to decide on the forecast. However, being the top selected feature, lagged_y hardly took up 10% of the importance weight, and the top 15 features only weighted 45% of the general importance. One way to interpret it was that a lot of features played a certain role at some point to help with the predictions, whereas it could also mean that the model found the predictors generally too plain to make any suggestion. Despite the concerns, the final LightGBM model successfully reduced the dimension of the dataset by 42 features.

Table 4 — Top 15 important features selected from the final LightGBM model.

Name	Top 15 Importance Score
lagged_y	0.09
lagged_AL_129	0.06
Clean_wt	0.05
COUCH VACUUM ACTUAL | 29:1P1420.PV	0.03

Name	Top 15 Importance Score
ENP TO DRYER SECT 1 DRAW | 31:DRAW003	0.03
PRI TOTAL HEAD | 31:IMV018	0.03
WIRE TO BINIP DRAW | 31:DRAW001	0.02
CW SCAN AVG | 31:F13SAN	0.02
REEL SPEED | 31:IMV005	0.02
SEC RUSH DRAG | 31:RDSM2	0.02
DRYER SECT 2 TO REEL DRAW | 31:DRAW005	0.02
ORTHO FLOW VACUUM 3 | 31:0PC2105	0.02
PM1 Total No.4 + No.5 Refiner KW Actual | 29:1J1458.PV-copy	0.02
PRI RUSH DRAG | 31:RDSM1	0.01
PRI ACID ADDITION RATE | 29:1F2001.RATE	0.01

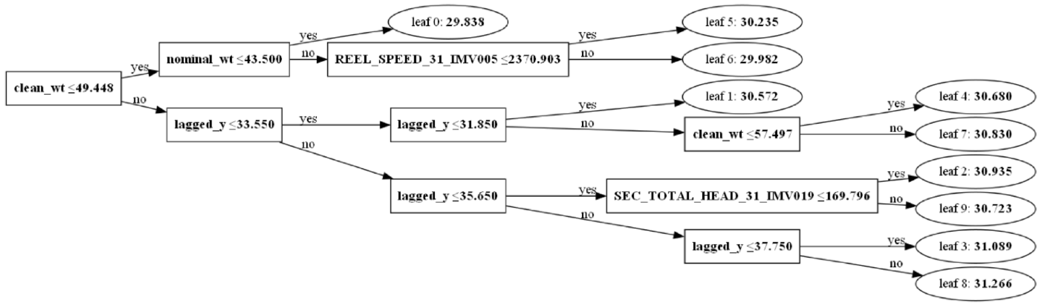

Figure 8—The first tree of the final LightGBM model.

Single-grade Model

There was also an attempt to compare the performance of LightGBM models in predicting all-grade and single-grade STFI results. Nonetheless, only 42#, HP56, and 69# were the grades that had a reasonable size of data to train and test on, and based on the RMSEs on the hold out set, there was no improvement found in the best of the single-grade models compared to the best all-grade model.

4.2.3 XGBoost

The next modeling method applied utilized the XGBoost package in python and was utilized in predicting the STFI target variable across all grades of linerboard as well as individual grades where there was enough data to accurately model and validate results. The all-grade model performed well during rolling window cross-validation. Numerous window sizes were tried while tuning the XGBoost's hyper-parameters. Interestingly, as the number of folds went up, the overall average Root Mean Squared Error (RMSE) of all the folds went down. Using 1,500 folds yielded average RMSE scores for all folds of 0.9726. The average Mean Absolute Scaled Error (MASE) could not be calculated due to only one record in each test fold. Interestingly, at 1,500 folds, each fold's test data only contained 1 record indicating XGBoost may have a good ability to utilize large amounts of data and predict the immediate next linerboard roll STFI tests with relatively good accuracy as measured by RMSE. This would, however, require the model to be re-trained in almost a real-time fashion and would likely not allow the company the opportunity to have any human validation of the continuously changing model parameters. Therefore, more folds were tested to better simulate real-world results.

Rolling Window Cross-Validation Results

Table 5 shows the various folds and RMSE and MASE scores for a defined set of folds attempted. As seen in Table 5, using 250 folds yielded a reasonable average RMSE of approximately 1.29 on the test data with an average MASE of approximately 0.89, indicating it performed better than the naïve model, using only the lagged target variable to predict future results. It also utilized 20 records for testing, allowing for a longer forecast window.

Table 5 — XGBoost cross-validation tuning results by fold.

Folds	Average RMSE	Average MASE	# Test Records per Fold
5	*1.59*	*1.03*	931
55	*1.40*	*0.92*	97
100	1.35	*0.90*	53
250	1.29	*0.89*	20

Folds	Average RMSE	Average MASE	# Test Records per Fold
500	1.22	*0.90*	9
750	1.17	*0.98*	5
1000	1.13	*1.17*	3
1500	0.973	*N/A*	1

Further tuning yielded the best results at 250-fold cross-validation utilizing the following XGBoost parameters: max_depth = 3, subsample = 0.8, colsample_bytree = 0.8, objective = reg: squared error, eval_metric = rmse, n_estimators = 100, learning_rate = 0.10 and random_state = 42. Setting the random state is optional but allows for the simple reproduction of results.

Final XGBoost Model Selection Process

The initial results were favorable, but the overall dimensionality of the data was still quite high, using 181 remaining features in the dataset. To reduce the dimensionality, each feature's importance was captured as well as how many times it appeared in the top 15 feature importance per fold of CV. This was then used to capture the top 15 features to use in a remodeling attempt to see how the reduction affected performance. The next round of modeling using only the top 15 features from the first round of cross-validation yielded an average RMSE of 1.3226 up from 1.2187 (8.5% increase) and an average MASE of 0.9279 up from 0.8963 (3.5%) increase. This was a small but expected increase of both measures.

To further simplify the model, the top 5 of these features were selected and another round of modeling and CV was performed. This resulted in an increase of average RMSE to 1.3455 and average MASE to 0.9499 or an overall increase of 10.4% and 6.0% respectively. The simplification of the model dimensionality yielded the most important features for the company and decreased model complexity and time to train the next model iteration. The top 15 features are described in Table 6. Table 6 highlights the importance of each feature selected in the top 15 and as the table shows, one feature, CW SCAN AVG | 31:F13SAN, had a much higher importance score than all the others. When the features were reduced to the top 5, this feature increased its importance. Figure 9 plots all 250 RMSE and MASE scores and shows a clear negative slope for each's trend line indicating with more data, XGBoost should be able to become more and more accurate.

Table 6 — Top features selected by XGBoost

Name	**Top 15 Importance Score**	**Top 5 Importance Score**
CW SCAN AVG | 31:F13SAN	*0.50*	*0.54*
PRI SLICE POSITION | 31:IMV022	*0.16*	*0.16*
lagged_y	*0.14*	*0.18*
REEL SPEED | 31:IMV005	*0.07*	*0.07*
PRI TOTAL HEAD | 31:IMV018	*0.05*	*0.05*
lagged_wt	*0.02*	*N/A*
F REF PLATE WEAR HOURS | 28:E6036D | Primary PLY	*0.02*	*N/A*
SEC TOTAL HEAD | 31:IMV019	*0.01*	*N/A*
lagged_AL_39	*0.01*	*N/A*
% FLOW RECYCLE	*0.01*	*N/A*
4G SHOWER FLOW | 28:GFC051.PV | Primary PLY	*0.01*	*N/A*
4G downleg vacuum | 28:Gv004 | Primary PLY	*0.01*	*N/A*
G Washer Vat Conductivity | 28:GCI1270 | Prima...	*0.01*	*N/A*
F REFINER POWER | 28:0JC083.PV | Primary PLY	*0.003*	*N/A*
ISO FLOW VACUUM 2 | 31:0PC2101	*0.003*	*N/A*

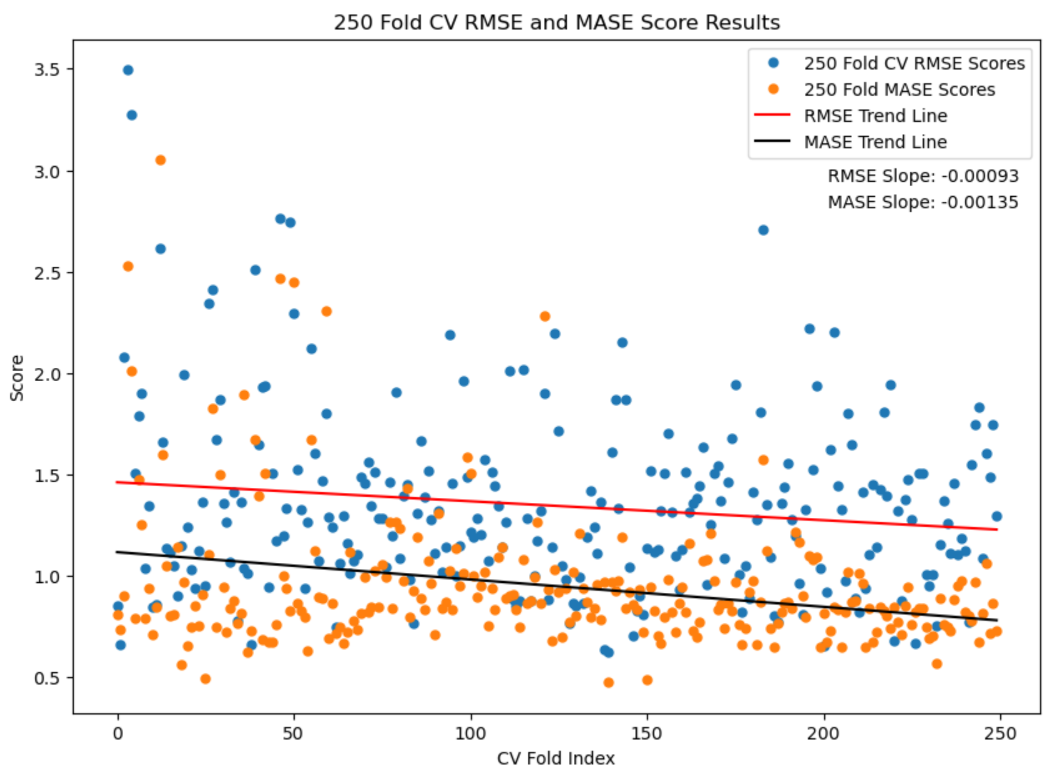

Figure 9— 250-fold RMSE and MASE score results plot showing a downward trend in both scores indicating better train/test performance over time.

Summary Results of Best XGBoost Model

The top 5 feature cross-validation yielded 250 separate models, and each was tested against the hold out data to see if it would perform better than a model trained with all available testing data. All models were tested against the validation/hold out set and the model with the lowest RMSE and MASE was number 48/250 with an average RMSE score of 1.8071 and MASE of 1.1226 yielding an increase of 34.3% in RMSE and 18.2% in MASE. This model performed slightly better than the model trained on all training data which resulted in an average RMSE of 1.8775 and MASE of 1.4852. Figure 10 shows a random sample tree provided by the chosen model #48.

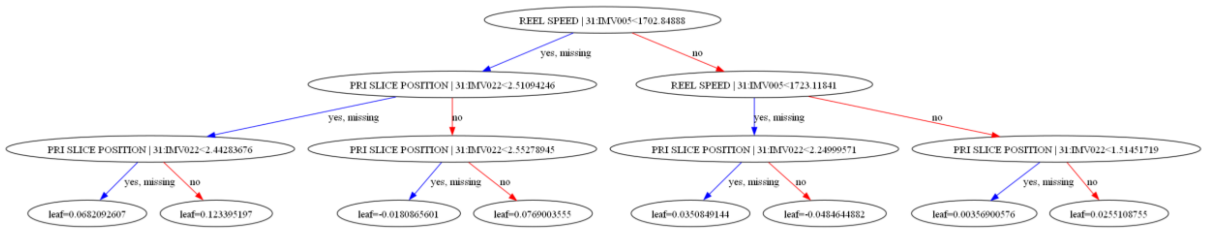
REEL SPEED | 31:IMV005<1702.84888
yes, missing
no
PRI SLICE POSITION | 31:IMV022<2.51094246
REEL SPEED | 31:IMV005<1723.11841
yes, missing
no
yes, missing
no
PRI SLICE POSITION | 31:IMV022<2.44283676
PRI SLICE POSITION | 31:IMV022<2.55278945
PRI SLICE POSITION | 31:IMV022<2.24999571
PRI SLICE POSITION | 31:IMV022<1.51451719
yes, missing
no
yes, missing
no
yes, missing
no
yes, missing
no
leaf=0.0682092607
leaf=0.123395197
leaf=-0.0180865601
leaf=0.0769003555
leaf=0.0350849144
leaf=-0.0484644882
leaf=0.00356900576
leaf=0.0255108755

Figure 10—The first tree of the final XGBoost model.

Single-Grade Modeling Attempt

XGBoost was also utilized to attempt to model in-grade STFI results and performed poorly due to lack of data for some grades and where there was enough data to model, RMSE scores on the hold out data were well above the threshold of < 2 RMSE after the model parameter tuning.

4.3 Model Comparison

Table 7 shows the key performance measures for each modeling method against the training and test data after cross-validation.

The training and test errors are shown for each model to indicate fit. Unlike the hold out errors, they are not comparable between models that use different cross -validation schemes. Because the noise level varies throughout the dataset, the training and testing errors of different schemes are disparately impacted by noise, since stretches of data are used and reused differently by different schemes. Only the first two models listed use the same cross-validation scheme and therefore their training and test error measures are comparable.

Of the linear models, the all-grade MLR performed the best. The single-grade model matched the hold out performance and therefore cannot justify the increased effort required to estimate it.

Table 7 — Final model performance comparisons

Model	**Avg. RMSE Train**	**Avg. RMSE Test**	**RMSE on Hold Out**	**Avg. MASE on Train**	**Avg. MASE on Test**	**MASE on Hold Out**
Naïve	1.57	1.57	1.70	1.00	1.00	1.00
Multiple Linear Regression (MLR)	1.46	1.49	1.59	0.91	0.93	0.93
MLR single grades	1.48	1.47	1.59	0.99	0.99	0.93
Random Forest	1.35	1.45	2.02	0.93	0.88	1.16
LightGBM	0.95	1.15	1.56	0.64	0.70	0.95
XGBoost	1.08	1.36	1.81	0.72	1.12	1.49

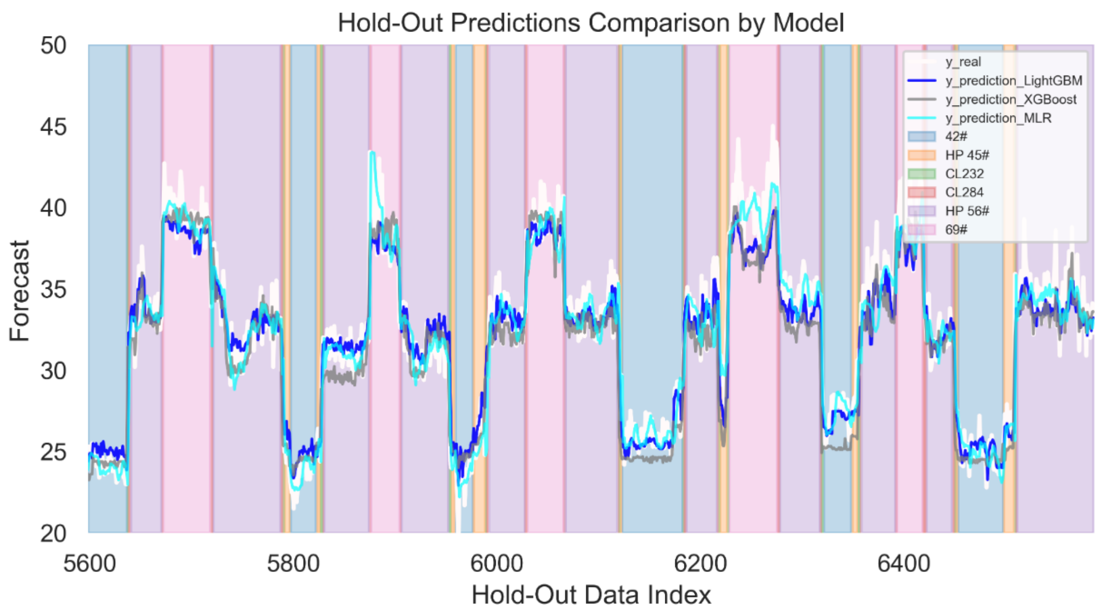

Figure 11—Hold out prediction comparison by final models

Figure 11 shows the comparison of final model predictions against the actual STFI test results. Each color in the background represents a different grade and each line represents model predictions and actual STFI test results.

Table 8 — Top non-sensor features selected by tree-based models

Feature Name	Selected by LightGBM	Selected by XGBoost
lagged_y	X	X
lagged_AL_129	X	
lagged_AL_39		X
Lagged_wt		X
Clean_wt	X	

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 Model Takeaways

As Table 7 shows, models which were trained using all linerboard grades performed within the 1-2 RMSE threshold set by International Paper on the hold out data. Something to consider would be the overall complexity of model implementation to reliably obtain results as well as the amount of effort required to retrain the models as time progresses. The LightGBM model marginally improved RMSE scores but that may not be enough to sacrifice the simplicity of using a linear model to make these predictions.

5.2 Recommendations

Each set of modeling methods should be evaluated long-term using multiple years of data and potentially over more than one machine to see if these methods could be improved further with more data diversity and time. If these models were to be used in a production environment, there would be a benefit to measuring these not only by RMSE but also by the cost to implement and maintain vs the cost savings realized from their predictions.

The tree models provided an interesting insight into what features are most important in providing accurate predictions. The features listed as most important in the models were relatively consistent throughout the tuning process, especially using XGBoost. Most of these features correspond to sensors built into the machinery and the company could put more focus on the maintenance of those sensors to ensure they are working and providing quality data for the chosen predictive model as shown in Table 9.

Table 9 — Important Sensor-Related Features Selected

Feature Name	Selected by Rnd. Forrest	Selected by LightGBM	Selected by XGBoost
REEL SPEED | 31:IMV005	X	X	X
PRI TOTAL HEAD | 31:IMV018	X	X	X
CW SCAN AVG | 31:F13SAN		X	X
PRI SLICE POSITION | 31:IMV022	X		X
SEC TOTAL HEAD | 31:IMV019	X		X
WIRE TO BINIP DRAW | 31:DRAW001	X	X	

5.3 Future Work on Topic

Retraining the complex models on multiple years of data may result in their exhibiting superior performance. Utilizing neural networks and other deep learning techniques, which are better at consuming large amounts of data, may also be worthwhile. These techniques may prove to be beneficial but would be complex to implement. Their performance would need to be much better than the techniques used in this paper or be able to reduce production costs in a significant way for the company to be worthwhile.

Weight is an important variable. There are several possibilities for improving it. Systematic evaluation of the arbitrary data cleaning rule used in this project may lead to a superior rule for weight. In addition, the current weight calculations only consider weights from half of the observations that include measurements of the target variable. Incorporating the eliminated observation's weights into the mean calculation may improve the power of the engineered weight feature, particularly for observations from early in a grade run.

If International Paper wants to work towards predicting the outcome of quality tests using only sensor data and input qualities, then a new MLR model using only such features could be compared to the Naïve model and the MLR model. The point is to assess the extent to which important parts of the process are invisible to the current collection of sensors. This sort of analysis can measure the effect of any additional sensors (or improved accuracy of current sensors). If new sensors were added, such a setup could be used to monitor the extent to which new sensors improve predictive power. A business goal of improving such predictions can be expressed in terms of MASE.

5.4 Lessons Learned

One of the lessons learned was about the ease with which the gradient boosting models would overfit the data due to the high dimensionality. Reduction of the number of features used to train the model improved overfitting using XGBoost but there still appears to be some level of overfitting that would benefit from further research and parameter turning.

The final LightGBM model could be improved. First, the criteria set to filter and sort for the best model might not be the optimal solution. There might be better rules to apply to the modeling outputs that are less prone to subjective judgment. Second, the tuning process was restricted by time and storage space. To get the most values out of the limited time and calculation power available to tune the models, the final model was only an outcome selected from a tuning process on four potentially important parameters (number of folds, maximum number of leaves, number of minimum data in a leaf, and fold index). In fact, other factors like tree depth, number of boosting iterations, number of trees in each model, and the maximum bin number can be good candidates to be included in the tuning process to achieve a better result. However, as more parameters come into place, as discussed in section 4.2.2, the modeling outcomes were difficult to interpret. A reason for the low interpretability is due to the boosted characteristics of the model. Another reason is that the connections were unclear among the metrics due to overfitting. The challenge remains to define the thresholds of parameters to avoid overfitting while maintaining the accuracy of RMSE on the hold out.

Another lesson involved the high variability of the hold out data, which was unexpected. The RMSEs of the Naïve model in Table 7 demonstrate the high level of noise in the hold out set. Although the additional noise lowered the performance of all models on hold out data, the hold out data was still legitimate production data. As such it serves to emphasize the highly variable nature of the underlying process.

6 REFERENCES

1. M.H. Waller. On-line Papermaking Sensors: An Historical Perspective. In The science of papermaking, Trans. of the XIIth Fund. Res. Symp. Oxford, 2001, (C.F. Baker, ed.), pp 785–895, FRC, Manchester, 2018. DOI: 10.15376/frc.2001.2.785.
2. Saraswat, M. (n.d.). Beginners Tutorial on XGBoost and Parameter Tuning in R. HackerEarth. Retrieved May 24, 2023, from https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/beginners-tutorial-on-xgboost-parameter-tuning-r/tutorial/
3. Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. Information Sciences, 191, 192–213. https://doi.org/10.1016/j.ins.2011.12.028
4. Staff, P. E. (2019, May 6). Cross-Validation strategies for Time Series forecasting [Tutorial]. Packt Hub. https://hub.packtpub.com/cross-validation-strategies-for-time-series-forec
5. Guolin K., Qi M., Thomas F., Taifeng W., Wei C., Weidong M., Qiwei Y., Tie-Yan L.. (2017, December). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
6. Shi, H. (2007, January 12). Best-first decision tree learning. Research Commons. https://researchcommons.waikato.ac.nz/handle/10289/2317
7. Wang, R., Liu, Y., Ye, X., Tang, Q., Gou, J., Huang, M., & Wen, Y. (2019, November). Power system transient stability assessment based on Bayesian Optimized LightGBM. https://www.researchgate.net/publication/340554211_Power_System_Transient_Stability_Assessment_Based_on_Bayesian_Optimized_LightGBM
8. L&W Autoline SCT ABB AbilityTM Quality Management System. (n.d.). Retrieved July 13, 2023, from https://library.e.abb.com/public/f58e80339d904231183d5de34839e2ebb/L&W%20Autoline%20SCT%20Datasheet_Finalv1.0.pdfAsdf

7 APPENDICES

7.1 Appendix 1

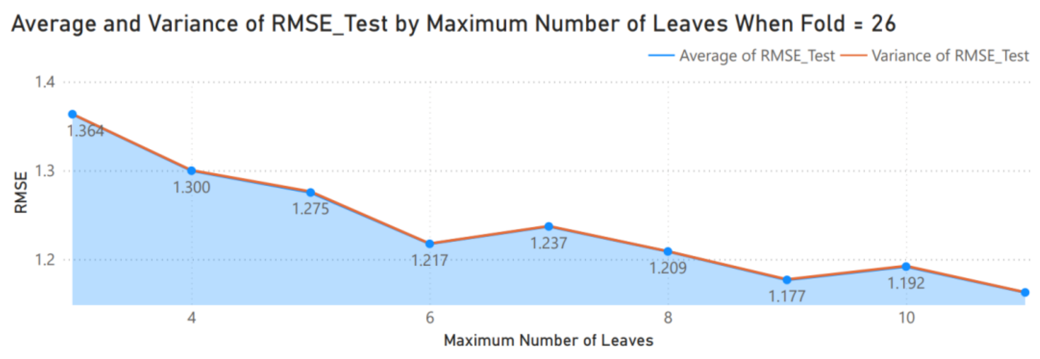

Figure 12—Average and variance of RMSE on test sets by the minimum data in a leaf when the number of folds equaled 26.

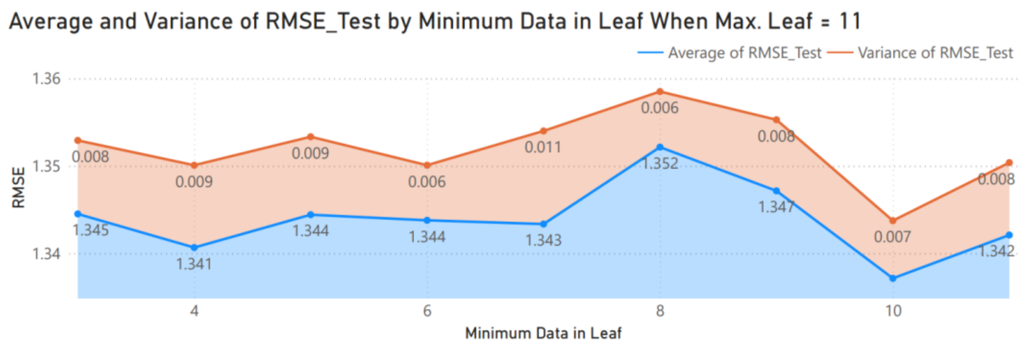

Figure 13—Average and variance of RMSE on test sets by the minimum data

7.2 Appendix 2

Table 10 — Important Sensor-Related Features Selected

Deleted Data Column Name	**Data type**	**Reason for Deletion**
DEFOAMER PP TO PM DAY TK | 29:0MSS050.PV	Categorical	Single valued
PM1 1 Wire/ 2 Saveall Defoamer Pump Stop/Run | 29:0MSS050B.PV	Categorical	Single valued
PM1 1/1A Saveall Defoamer Pump Stop/Run | 29:0MSS050A.PV	Categorical	Single valued
GRADE CODE | 31:GRADE.STRING	Categorical	Redundant
INDEX | REEL ID NUMBER	Categorical	Redundant
Pine 1 Scan Residual EA | 27:0P1RESEA1.MN | Primary PLY	Numerical	Single valued

Deleted Data Column Name	Data type	Reason for Deletion
Pine 2 Scan Residual EA ∣ 27:0P2RESEA2.MN ∣ Primary PLY'	Numerical	Single valued
SAVEALL #1/1A DEFOAMER PUMP SPEED ∣ 29:0HS050A	Numerical	Single valued
WIRE 1/ SAVEALL 2 DEFOAMER PUMP SPEED ∣ 29:0HS050B	Numerical	Single valued
PM1 1st Press Air Bag Loading, Back ∣ 4296	Numerical	> 10,000 missing values
PM1 1st Press Air Bag Loading, Front ∣ 4295	Numerical	> 10,000 missing values
PM1 1st Press Panel Air Reading, Back ∣ 4298	Numerical	> 10,000 missing values
PM1 1st Press Panel Air Reading, Front ∣ 4297	Numerical	> 10,000 missing values
PM1 2nd Press Air Bag Loading, Back ∣ 4300	Numerical	> 10,000 missing values
PM1 2nd Press Air Bag Loading, Front ∣ 4299	Numerical	> 10,000 missing values
PM1 2nd Press Panel Air Reading, Back ∣ 4302	Numerical	> 10,000 missing values
PM1 2nd Press Panel Air Reading, Front ∣ 4301	Numerical	> 10,000 missing values
PM1 ENP Loading ∣ 4305	Numerical	> 10,000 missing values
PM1 Lump Breaker Roll Air Bag Loading, Back ∣ 4304	Numerical	> 10,000 missing values
PM1 Lump Breaker Roll Air Bag Loading, Front ∣ 4303	Numerical	> 10,000 missing values
PM1 Primary Freeness ∣ 2951	Numerical	> 10,000 missing values
PM1 Primary Headbox Consistency ∣ 2950	Numerical	> 10,000 missing values
PM1 Secondary Freeness ∣ 2956	Numerical	> 10,000 missing values
PM1 Secondary Headbox Consistency ∣ 2948	Numerical	> 10,000 missing values
REFINER #1 OUTLET PRESS ∣ 29:1P1442	Numerical	> 10,000 missing values

7.3 Appendix 3

The Naïve model sets the coefficient on the lagged target value to unity by assumption. A Naïve OLS model, involving only the lagged target value, is used to report the errors. Since the OLS-generated coefficients were within 0.0001 for all estimations calculated with the project data, distinctions between the two models are ignored