Data Science Internship Experience at Panasonic Energy of North America

Nicholas Ward, Supervisor Signature (George Savariar):

August 26, 2023

Abstract

This paper will go in-depth on the experience that was gained at Panasonic Energy of North America as a Data Scientist. There were two projects worked on, one was incomplete as the project was going to take more time to complete than the 12 weeks. The second project involved Natural Language Processing (NLP) to see how much value that could be derived from operator input and if it was possible to predict machine downtimes from text input from those operators.

1 Introduction

The objective of the first project was to analyze battery top crimp defects (mainly scratches on the top crimp of batteries). The idea was to try and find any correlations between the top crimp defects and machines. That way modeling could be used to predict where the problems were arising right when the top crimp defect was detected by the AI Camera. This project took up two weeks of the time at the internship but the idea was deleted when Michael Atkinson (Supervisor) decided it was going to take too long for reasons outside of anyone's control.

The second project was the one that took up the rest of the time at the internship. This was an NLP project. NLP is how machines interact with human language and is a way to extract something useful out of human language. The dataset that was used for this project was pulled from the slitting department. Slitting is where the Cathode and Anode portions of the battery get cut into smaller pieces to go from one large piece of metal down to one that will fit inside the battery casing. The goals of this project:

(i) Accurately forecast down times.

(ii) Cluster Actions Taken or Description of Issue into meaningful clusters.

(iii) Use those clusters to improve down-times forecasts.

(iv) Use text analysis to determine potential drop-downs for actions taken and description of issues.

(v) Build a recommender that can list actions and operator can take and their predicted down-times given a description of the issue an other features.

(vi) Build a Flask app to show how this project would work in the factory.

2 Technology Stack

The NLP project required multiple different technologies to complete. The most important was the python language. This was used in almost every facet of the project. Python is a powerful language for data analysis, data manipulation, and building machine learning models.

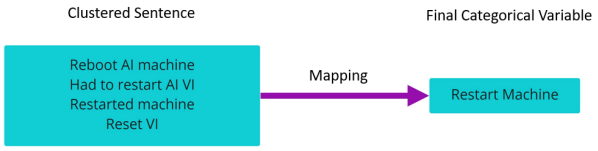

Figure 1: An example of what NLP clustering will accomplish is mapping similar sentences to a final categorical variable.

To be able to pull data from PENA's server MySQL was used. MySQL is a relational database management system developed by the company Oracle that is based on structured query language. MySQL allows for multiple different commands to pull the exact data that is wanted from a rather large database. This makes it easier to begin data manipulation with python when done correctly to optimize server-side data pulls.

For exploratory data analysis, Jupyter was a helpful tool for the quick and efficient creation of graphs and tables. Jupyter is split into cells that make running sections of code much easier than normal. With the graphs and tables printed out right below the cells allow for quick visuals as well.

Anaconda was used to manage dependencies. When doing projects on the main files, version issues quickly arise and it becomes difficult to manage them. To save massive headaches in the future, Anaconda is quick and easy to understand, and manages different environments for different projects taking away any worries of dependency version issues.

Data manipulation was entirely done by pandas. Pandas is one of the best libraries to have data in what is called a Data-frame. A Data-frame has rows and columns and conveniently maintains data. Using pandas allows for hundreds of manipulation techniques.

To create machine learning models, Scikit Learn was used. This is a rich library that contains multiple different functions to help make building machine learning models as easy and pain-free as possible.

Flask/ HTML was needed to create a demo app to show the Panasonic Technology Group (PTG) during the presentation. The demo was a way to show how the project would function in production.

Last but not least to manage edits and versions of code for this project, GitLab is used. GitLab is the company's choice of managing repositories. This makes it easy to keep track of code changes, issues that might arise, and other people can see them (such as my supervisors).

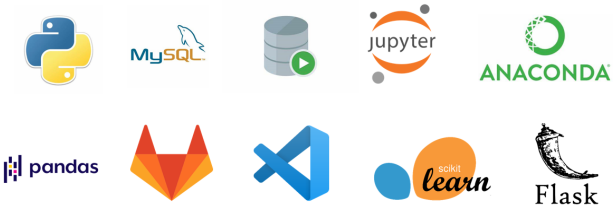

Figure 2: The techstack that I used throughout this NLP project

3 Slitting Machine Downtimes Dataset

The dataset that was used throughout the project was pulled from Jira software (a project management software but can be a place to store some data). The Slitting teams project management implemented a system where operators logged some information when a machine in slitting went down. When an incident occurs that causes downtime, the operator logs, the simple root cause, machine numbers, start downtime, end downtime, process (Anode or Cathode), action taken (Natural language), and description of the issue (Natural language).

The full data set had 54,000 records, and upwards of 29 simple root causes. To be able to analyze all 29 simple root causes would have been impossible for the short 12-week span that the internship was going to last. That's why only five simple root causes were taken into

account to make this project a proof of concept. Almost like a proposal that this project is worth putting more time into. The five simple root causes that were used in this project were: General Quality Issue, AI VI Issue, Blade Exchange, Side Edge, and Material Break. The decision on which simple root causes would have the most impact on the company was made by the slitting team. After an hour-long discussion the team and I had narrowed down the simple root causes to the above five and those were the ones I would stick with for the rest of the internship.

4 Exploratory Data Analysis

Exploratory Data Analysis is essential to truly understand the problem at hand. I was able to look at the different features and how they relate to the down-times. In Figure 3 the down-

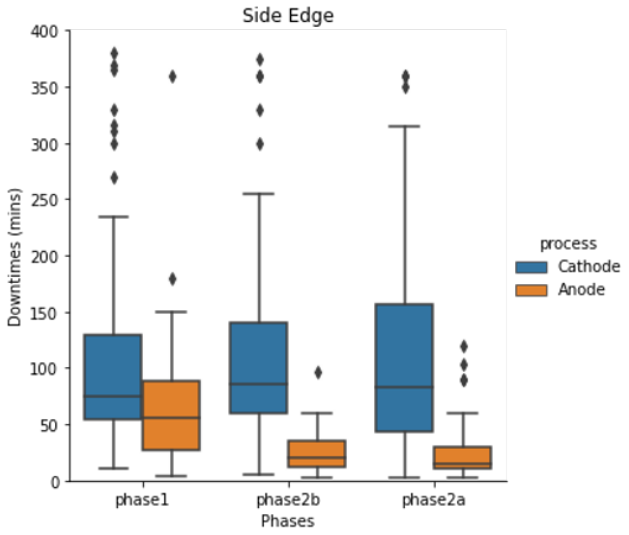

Figure 3: Downtime durations based on the phases (phase 1, phase2b, and phase 2a) as well as downtime durations based on the process the downtimes happend (Cathode or Anode).

time durations are split, where if a downtime happened in the Cathode process no matter what phase the downtime occurred in the minutes can be expected to be higher than that of the Anode process. Figure 3 was created from the simple root cause side edge. This means that process could be a good feature to use in the model to hopefully increase model performance.

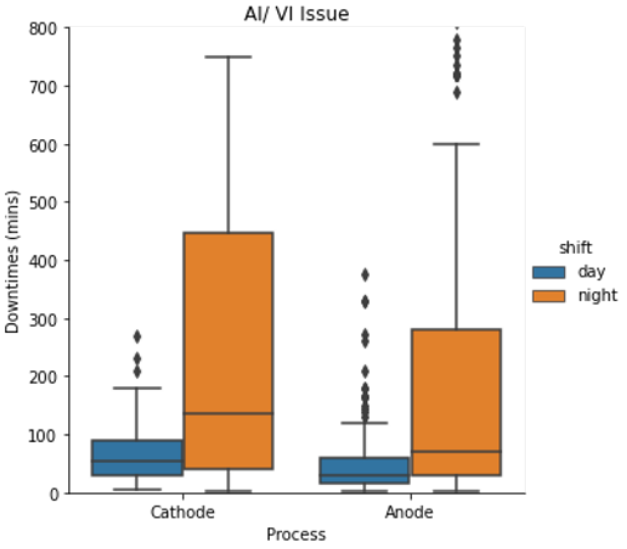

Figure 4: Downtime durations based on the process (Cathode or Anode) as well as downtime durations based on the shift (Day or Night) that the downtime happened. The simple root cause used in this figure was AI/ VI Issue

Figure 4 shows that the downtimes can be expected to be higher during the night shift on either Cathode or Anode processes. This means that for AI/ VI Issue we can expect that shift will have an effect on the model performance.

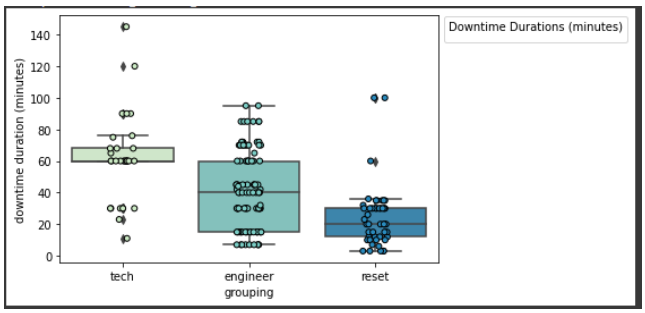

Figure 5: Downtime durations based on certain words that show up in sentences for simple root cause AI/ VI Issue.

The final example I wanted to show was looking at words that might show up often in different simple root causes. Then if that word showed up, the downtime would be logged and added to a graph to showcase the downtimes. Figure 5 had words, tech, engineering grouping, and reset. There is a clear difference in downtimes from just certain words that show up. Where in the Figure 5 example we have sentences that have reset in them that will have on average lower downtimes. But when an engineer or tech lead needs to come out and help an operator, the downtimes are larger. This makes complete sense and is reassuring that it's showing up in the data.

The first couple of weeks of this project was purely EDA. My mentor would not allow me to move on to any modeling or data cleaning until I had a full understanding of how each feature interacts with each other and how they affect downtimes. I only showed three to save time and space on this project, but I had down hundreds of these graphs to get the optimal features for each simple root cause in question.

5 Feature Engineering

The original data from the slitting team is quite useless without a little bit of feature engineering. The first step was to create a target variable. In the original dataset, there are only columns for the start of downtimes and the end of the down-times. The target variable is going to be the log of the actual downtimes, which means start and end downtimes are subtracted from each other and then the log is taken with that. Taking the log of the target scales down the values increase the prediction accuracy later in the models.

The main features that were created to go into training were a phase, machine line, machine number, operator shift, and topic clusters that are going to be explained in the NMF section of this paper. The phase is found by taking the first number in a full machine number (Example machine full number: 8-2). Where if the first number is 1 or 2 then it falls under the phase 1 category. Numbers 3, 4, and 5 are in phase 2a. Numbers 6, 7, or 8 are in phase 2b. The machine line itself is also just the first number of the machine's full number. So an example would be if the full machine number is 8-2 then the line number would be 8. For the machine number, it is the last number in the full machine number. An example of the machine number would be if it is 8-2 again then the machine number is 2. The last feature that was derived from the original dataset was that of operator shift. It was considered the day shift if the start downtime was between the hours 07:00 and 19:00 and then considered night time if the time falls outside of the daytime range.

6 Natural Language Processing

Natural Language Processing (NLP) is how machines interact with human language. It also can be thought of as a way to extract something useful out of human language. There are quite a few different ways to get something useful out of human language using NLP. The steps to complete this NLP project were:

(i) Clean text data.

(ii) Vectorize sentences using Term Frequency - Inverse Document Frequency.

(iii) Cluster sentences into topics based on similarity using Non-Negative Matrix Factorization.

6.1 Cleaning Text Data:

Cleaning the text data is straightforward. The first few steps required are to lower case all words, delete punctuation ([,.;:'], etc), delete all numbers (0-9), delete special characters (@, , !, etc), replace synonyms with one common

Figure 6: An example of NMF clusters from the TF-IDF matrix. The left graph is the clustering for the action taken. There were 30 clusters that NMF had made. The right graph was the clustering for the description of issue. This one had only 20 clusters.

word ('reboot', 'reset', 'restart': all changed to 'restart'). Then removing stop words is important. These are words that might show up often in the corpus and add no uniqueness and confuse the clustering algorithms. Examples of stop words are "the", "a", "I", and "in".

Then stemming each word in the sentence. Stemming refers to the process of reducing a word to its word stem that affixes to suffixes and prefixes or the roots. Lastly, the words are lemmatized. The word "Lemmatization" is itself made of the base word "Lemma". In Linguistics (a field of study on which NLP is based) a lemma is a meaningful base word or a root word that forms the basis for other words. For example, the lemma of the words "playing" and "played" is "play". The final step is to tokenize each sentence to prepare for vectorization. This mean to change a sentence like (["hello there how are you"] to ["hello", "there", "how", "are", "you"]). This way we can develop a corpus and assign a weight to each word to determine its unique value.

6.2 TF-IDF:

Term Frequency - Inverse Document Frequency (TF-IDF)is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. The larger the weight a word is in a document the more important that word is to the document. TF-IDF will assign a weight to each word in the corpus to where a dataframe can be created where the rows are the documents and the columns are the words. That way there is a nice way to store and use the TF-IDF values for a document. A sentence: "Hello my name as nick" would be represented like [0.05, 0.0, 0.65, 0.0, 1.23] in the row where each column represents a word in the example sentence. The equation for TF-IDF is:

$$TF - IDF = TF_t * log\frac{N}{DF_t} \qquad (1)$$

Where TF_t is the term frequency of the term t. (how many times does the term occur in the document?). N is the total number of documents in the corpus. DF_t is the document frequency of t (How many documents have the

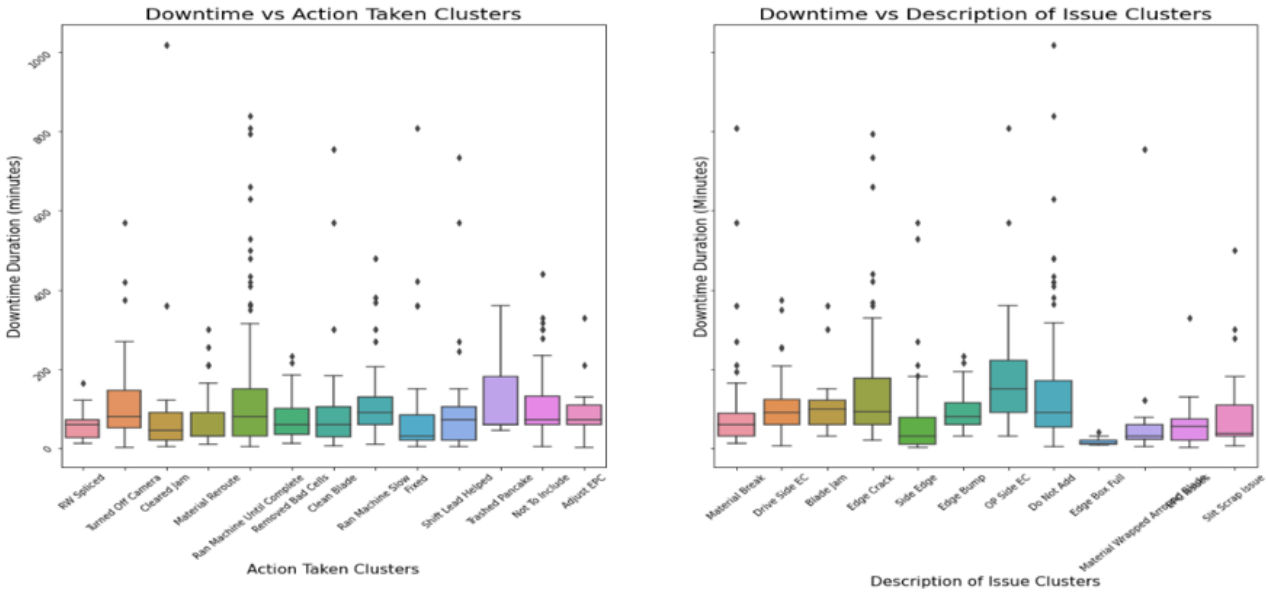

Figure 7: An example of improved clustering using a rules based method and mapping to new understandable categorical variables

term t in it?). $log\frac{N}{DF_t}$ is the Inverse Document Frequency (IDF) portion of TF-IDF.

6.3 NMF:

Non-Negative Matrix Factorization (NMF) is the technique that is used to cluster the documents based on similarity. It is an unsupervised machine learning model. How NMF works, consider we have an input matrix T of shape m x n. This method factorizes T into two matrices W and H, such that the dimension of W is m x k and H is n x k. For our situation, T represents the TF-IDF matrix, each row of matrix H is a word embedding and each column of matrix W represents the weight of each word in each sentence. All the entries of W and H must be positive given that all the entries of the TF-IDF matrix are positive.

In the case of this project, we have a dataset of operator inputs, a description of the issue, and action is taken. In the TF-IDF matrix (input matrix), we have individual documents along the rows of the matrix and each unique term along the columns. If we see a description of an issue such as reboot computer, AI decided to reboot, or operator rebooted computer it may be grouped under the topic AIreboot. In this method, each of the individual words in the document term matrix is taken into account. While factorizing, each of the words is given a weight based on the relationship between the words. But the one with the highest weight is considered as the topic for a set of words. So this process is a weighted sum of different words present in the documents.

NMF was run on all five simple root causes, in Figure 8 was the simple root cause "Side Edge". All simple root causes were run where for a description of an issue the factorization was reduced to 20 topics and action taken was clustered into 30 topics. This was so that all relationships would be captured as accurately as possible.

Looking at Figure 9 we can see exactly what a sentence cluster looks like from NMF. NMF does a great job at clustering similar sentences, especially when the number of clusters is in the higher 20/ 30 range.

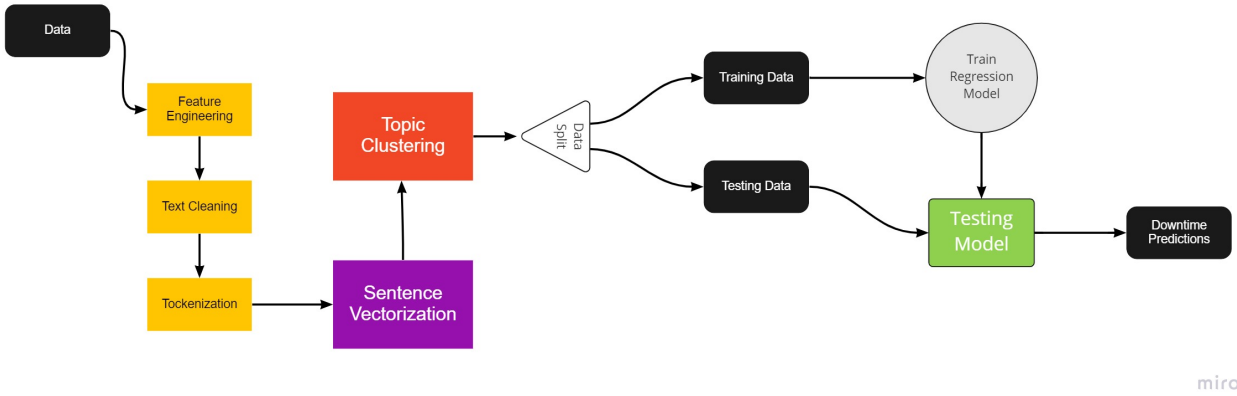

Figure 8: The model flow, from data preprocessing all the way to the predictions for training.

```
22     operator side edge cracks causing delays due t...
38     operator side edge cracks causing delays due t...
44     slight dr side edge cracks causing delays due ...
49     35 min delay.  intermittently punching for edg...
59     edge bump.  edge crack on the operator side. i...
73     intermittently stopping to punch edge crack c0...
93     occasional drive side edge crack's causing del...
99     operator side edge cracks causing delays due t...
100    operator side edge cracks causing delays due t...
103    operator side edge cracks causing delays due t...
108                  stopping to punch edge crack defects
153    operator side edge cracks causing delays due t...
154    operator side edge cracks causing delays due t...
155    minor operator side edge cracks causing delays...
157                        edge cracks stopping to punch
160    operator side edge cracks causing delays due t...
163    1 hour delay, intermittently punching edge crack
172           intermittently punching edge crack defects
200    drive side edge cracks causing delays due to s...
348    #10 punching every few cells, wouldn't clear u...
359    vacuum system stopped operating.  no leaks in ...
374    new material received has severe edge crack an...
380    edge crack on material kept machine stopping a...
395    edge crack and some defects kept machine stopp...
408    op side edge cracks. \r\nslowing down often to...
532    coil with edge cracks on op side of coil causi...
544    heavy defects and bad edge cracks kept the mac...
566                    running slow---severe egde cracks
```

Figure 9: Here is an example of what inside a topic cluster would look like. This example was created from "Side Edge" simple root cause.

7 Rules Based Method

A rules-based method was introduced to further improve the clustering accuracy. The downfall with such a large amount of clustering as seen in Figure 8 is that there starts to become multiple clusters that have different sentence structures but mean the same thing. NMF doesn't understand this but we humans do. That is where a rules-based system comes into place. I would look at every cluster and take not what the general topic was in that cluster. Then I would find other similar topics and merge them.

8 Model Forecasting

Once the data has been cleaned and clustered correctly, it can now be used for input to a regression model. To test models accurately K-folds cross-validation was used. The testing involved five folds and the idea behind k-folds cross-validation is that the models can be validated on multiple different sets. It tests to see if the model is overfitting the training data. That means that with five folds the models will see five new sets. This shows the best results in terms of testing models.

After all the models were tested using K-folds cross-validation, the highest performing model was the ensemble model (the bold values in table 1 that uses, LightGB, XGBoost, Lasso, and Lassolars).

8.1 Results:

In Table 2 shows the different simple root causes and their MAE score with NLP, without NLP, and a bench mark of taking the mean

Table 1: K-Folds Cross Validation scores on Side Edge with multiple different regression models. The scores are the log of the downtime duration. STD stands for standard deviation.

Model	**Score (mean)**	**STD**
Random Forest	0.8099	0.1242
Lasso	0.8483	0.0996
Ridge	0.8519	0.1015
Ensemble	**0.8066**	**0.1148**
Lassolar	0.8406	0.1025
Neural Network	0.9585	0.0900
Elastic	0.8485	0.0998
GBoost	0.9232	0.0891
LightGB	0.8611	0.1128
LightGB	0.8611	0.1128
XGBoost	0.8366	0.1154
KRR	0.8506	0.0901

of the validation data set. These values were calculated during the training and validation of the model. What is important to note is that the MAE with NLP is lower than MAE without NLP and lower then the benchmark. This means that NLP really does make a difference in our predictions which is great.

Table 2: All the values are in minutes. This shows the mean absolute error (MAE) of the model with NLP, with out NLP, and a benchmark by taking the mean of the validation set.

Simple Root Cause	**MAE**	**MAE no nlp**	**MAE (mean)**
General Quality Issue	76.944	82.733	94.033
Material Break	46.245	48.638	50.965
Blade Exchange	59.997	65.088	66.041
Side Edge	63.849	72.275	79.318
AI/ VI Issue	92.372	104.321	134.132

Some more evidence that NLP clusters do make a difference in our model's performance can be seen in Figure 13. It shows the residuals of all five of the simple root causes in question. You can see that the residuals for NLP models is tighter of a spread than that of the no NLP models as well as the mean benchmark.

Figure 19 shows the scatter plots of the

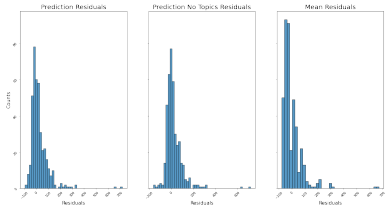

Figure 10: Material Break

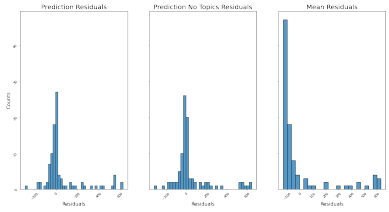

Figure 11: AI/ VI Issue

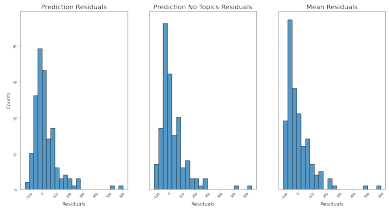

Figure 12: Blade Exchange

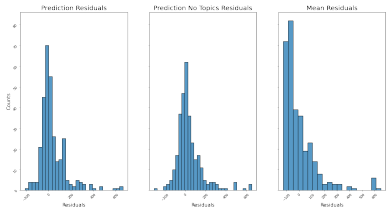

Figure 13: General Quality Issue

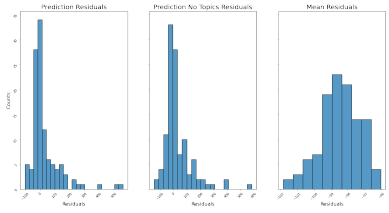

Figure 14: Side Edge

Figure 15: Prediction Residuals for all five simple root causes. Left is the residuals with NLP, middle graph is the residuals without NLP, and the right graph is the mean benchmark

model predictions vs the actual values. We can see that when you draw a perfect linear line through those predictions that it cuts the data

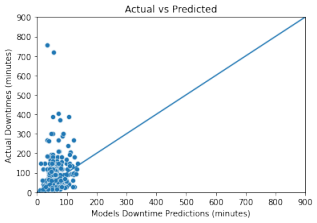

Figure 16: Material Break

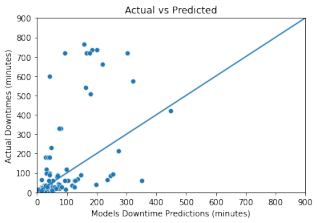

Figure 17: AI/ VI Issue

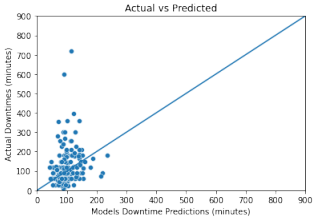

Figure 18: Blade Exchange

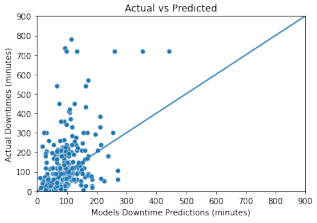

Figure 19: General Quality Issue

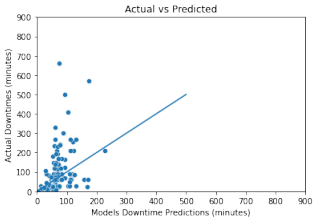

Figure 20: Side Edge

Figure 21: Scatter plots of model predictions vs. actual downtimes.

in half. This shows that the ensemble model is not biased and under/over predicted the data.

9 Flask App / Recommendation System

To show a proof of concept on how models can be deployed in production I created a flask app. Flask used with HTML can create nice web pages that can be hosted on the PENA server when it is ready to be used in production. The

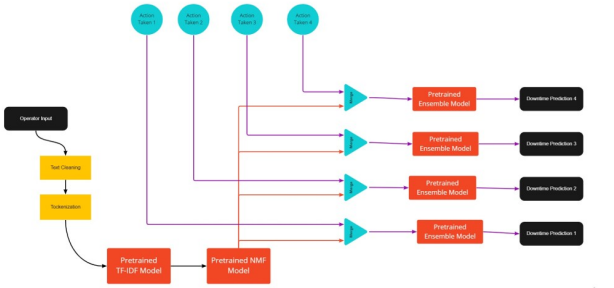

Figure 22: Recommendation flow for the proof of concept flask app to recommend actions to fix issues and the predicted downtimes.

recommendation system works by adding loading in the pre-trained models, NMF, TF-IDF, and an ensemble model. The operator will then input a description of the issue which will then go through all text cleaning and tokenization steps. Then once the description of the issue has been done that it will go through the pre-trained models TF-IDF and NMF. Then an action taken topic will be merged with the data that comes out of NMF. We can run each action taken merge data into a separate model and predict downtime durations for each action taken.

This will provide insight to the operators during production as when downtime occurs they will not have to think about actions to take to fix the machines. The app will simply tell them what the best actions to take are and the expected downtime of those machines. Then it is up to the operator to decide what action to take.

10 Conclusion

To improve the results of the NLP project would be mainly to fix how the operator inputs data. I realized early on how dirty this data was due to operator error. The first and most helpful fix would be to set a character minimum to the description of issue and action taken. Some of the input was in one-word sentences, such as "fixed" or "completed". These have no meaning to humans and especially have no meaning to clustering algorithms. Forcing operators to write more would increase cluster accuracy immensely. A few more fixes would be, no negative downtimes allowed, a spell checker implemented in the system, and no downtimes greater than the operator shift lengths. One issue that was missed during implementation was taking into account possible breaks or lunches that operators take that increase the overall downtime due to operators not inputting the fixed downtimes until they get back from their breaks. Having a section to include that asks if they took a break or not would help as well.

This internship turned out a lot better than I expected. My mentor was extremely helpful throughout the experience, always ready to answer any questions that I had. Panasonic made it their mission to give the interns projects that they would enjoy. And that was the case for my project. I enjoyed every minute of it.

One of the most important things I learned that my mentor drilled into my brain was limiting artificial stupidity (Throwing data into a model hoping it does something for you.). Data quality is more important than algorithms that do the predictions.