Reducing Manufacturing Failures

Aayush Mudgal
Computer Science
Columbia University
am4590@columbia.edu

Sheallika Singh
Data Science
Columbia University
ss5136@columbia.edu

Vibhuti Mahajan
Data Science
Columbia University
vm2486@columbia.edu

***Abstract*—It is of high importance for big companies to maintain high quality of products. In this project we use the measurements of the parts of the appliances as they progress through an assembly- line to predict whether there would be a defect in the part. This will help companies to produce high quality, low cost products at the user end. Building upon inferences from data and an MCC metric to evaluate test-performance, we present a workflow for predicting the machine failures using classification techniques such as random forests and gradient boosting.**

Keywords-assembly-line; MCC; random-forests; gradient-boosting;

I. Introduction

Big manufacturing companies like Bosch, have an imperative to ensure that the production is of high quality. Part of doing so is closely monitoring its parts as they progress through the manufacturing processes. Since Bosch records data at every step along its assembly lines, they have the ability to apply advanced analytics to improve these manufacturing processes. However, the intricacies of the data and complexities of the production line pose problems for current methods.

Manufacturing companies are faced with the task of predicting the internal failures using thousands of measurements and tests made for each component along the assembly line. This would enable the production company to bring quality products at lower costs to the end user (by minimizing human interventions).

The challenge is posed by the intricacy of the production line data. Here we deal with a huge number of features with highly skewed response variable as precisely we have 171 correct samples for every failure. Other than that we also have to deal with multiple values for categorical features, large number of missing values and a lot of duplicate values. This required careful and repeated experiments during pre-processing so that no information is overlooked while presenting the findings of our model and also to make sure that the model is computationally feasible for the manufacturing firm to use over and over again.

In this report we'll first demonstrate some interesting findings that we came across while playing with data and that later became useful while building the actual model, then we'll progressively build the model giving our insights at every step and then finally we'll conclude with our results and possible extensions of this project.

*please find larger and clearer images in Appendix

II. Related Works/ Findings

While deciding the statistics to create an error prediction model using the given dataset, we should keep in mind that the response variable is going to be either fail/pass or possibly the number of failures in a time period. Logistic Regression seems to be the most intuitive choice in this case. However the independent variables are time dependent and although there isn't any autocorrelation in the data, errors that do occur tend to occur close together.

The literature developed by the "Six Sigma[10]" school of manufacturing quality control is probably the single, best source for applied significance testing of defects, for looking into predicting product defects from on-line time series data. The statistical representation of Six Sigma describes quantitatively how a process is performing. To achieve Six Sigma, a process must not produce more than 3.4 defects per million opportunities. A Six Sigma defect is defined as anything outside of customer specifications. Six Sigma is a measurement-based strategy for process improvement and problem reduction completed through the application of improvement projects. This is accomplished through the use of two Six Sigma models: DMAIC and DMADV.

- DMAIC (define, measure, analyze, improve, control) is an improvement system for existing processes falling below specification and looking for incremental improvement.
- DMADV (define, measure, analyze, design, verify) is an improvement system used to develop new processor products at Six Sigma quality levels.

III. System Overview

Bosch, provides a huge dataset[1], representing the measurements of parts as they move through their production lines. The goal is to predict whether a particular part (identified by a unique Id) will fail quality control or

[1] kaggle.com/c/bosch-production-line-performance/data

not. For better storage and easier understanding of this huge data, the data is separated into different files based on the type of the feature they contain: namely numerical, categorical and date features. Features are named such that it depicts the production line, the station and the corresponding feature number. For example: the feature L3_S36_F3939, depicts the feature measured on line 3 at station 36 and the feature number is 3939. A general overview of the datasets sizes is shown in the following table:

	File Size (Zipped)	File Size (Unzipped)	Rows	Column
Categorical (Train/Test)	20 MB	2.50 GB	118378	2141
Numeric (Train/Test)	270 MB	2 GB	118378	1157
Date (Train/Test)	59 MB	2.7 GB	118378	970

It is a very challenging dataset because the ground truth is highly imbalanced and also it is among the largest datasets (in terms of the number of features) ever hosted on Kaggle.

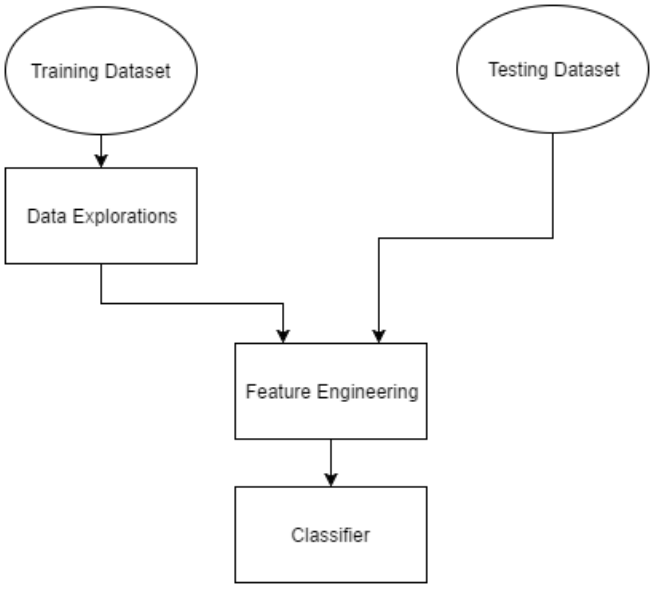


Figure: General System Overview

IV. Evaluation Metric

Since the dataset is highly imbalanced with respect to the response variable, (171 correct samples for every failure), a simple 0-1 loss would be meaningless. We use the Matthews Correlation Coefficient (MCC)[1], [2] as a measure of the quality of the binary classifiers. MCC is basically a correlation coefficient between the predicted and observed classes. A coefficient of +1 represents a perfect prediction, coefficient of 0 represents no better than random prediction, while a coefficient of -1 signifies a total disagreement between the predicted and observed values.

$$MCC = \frac{(TP*TN)-(FP*FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

where TP stands for True Positives, TN stands for True Negatives, FP stands for False Positives, FN stands for False Negatives.

V. Algorithm

The training dataset is quite huge in itself, with a very large number of features, thus it is very important to carefully explore and understand the dataset. Since no prior information about the features is known, we try to find relevant relations between the features after exploring for interesting informations in the dataset. After having gained insights into the dataset, we chose the best set of features that explain the dataset the most, and work on building our classifier on the same.

A. Data Exploration

We tried to find relations between the different features, the stations they are recorded on, the production line the station is on, and the response variable corresponding to these observations to understand which features, stations and lines play more significant role than the others. Some of the interesting dataset explorations are recorded here in this report. Through Figure 1 and 2, we observe the frequency of different features corresponding to the stations. We observe that some stations (namely Station 24, 25, 29, 30) have a very high number of features corresponding to them, while they are many stations that have very low number of features corresponding to them. These results do suggest some relative importance of stations in identifying the stations, which perhaps would play, a more important role in deciding the response variable.

*please find larger and clearer images in Appendix

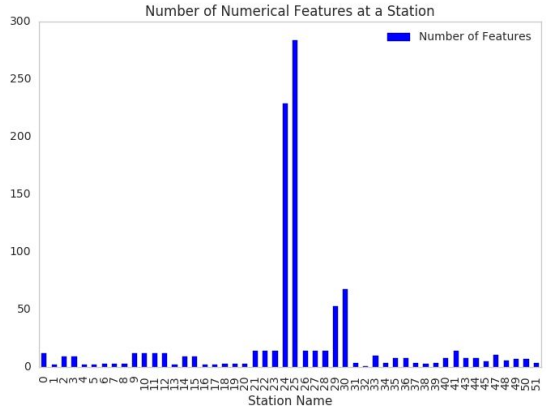


Figure 1

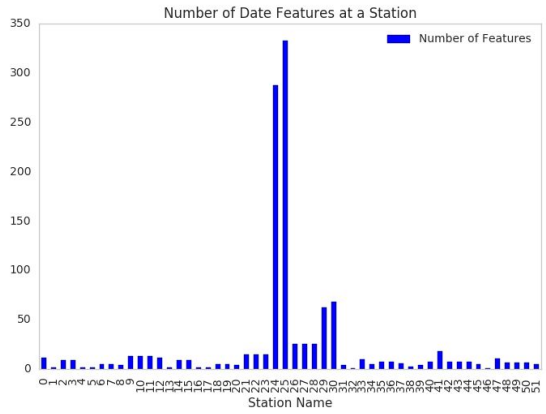
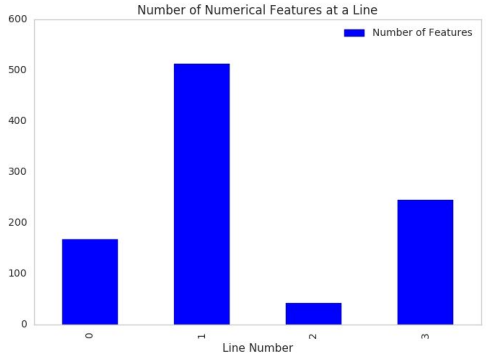


Figure 2
*Figure: Frequency of Features vs Station Number

We report similar trends for features with respect to different lines (Figure 3 and 4), suggesting the relative importance of one line over the other in deciding the response variable. It could be interpreted logically that when a product is probable to be defective, it is shifted to a line where it undergoes more strict manufacturing tests. Thus having more number of features being recorded for it.



*please find larger and clearer images in Appendix

Figure 3

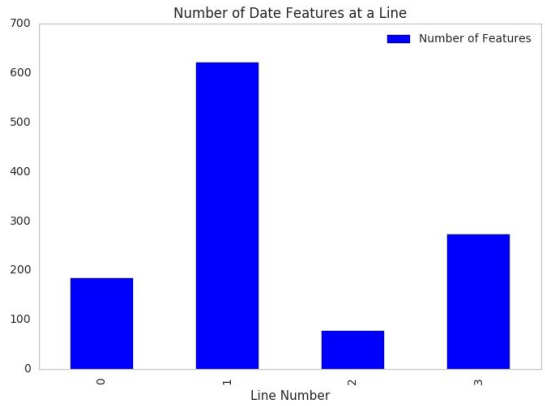


Figure 4

In the Figure 5, we visualize the first 1000 jobs as they progress through stations over the production lines. The bigger the size of the node, the more the number of jobs come into that station while similar colour nodes signify that they lie on the same production line. From this visualization we can get a rough intuition that the lines are somewhat parallel and there seems to be some backup/overflow lines.

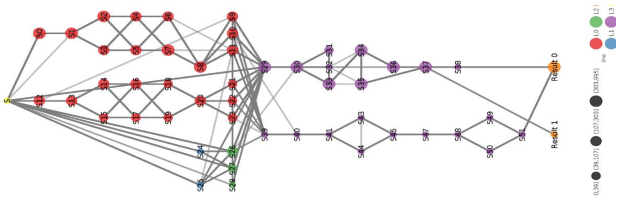


Figure 5
*Figure: A graphical representation of the first 1000 jobs

In Figure 6, we visualize the progress of a job over time as it goes through different processing times. Colour represents the line number the job is on at a particular instant of times. The first two rows corresponds to 8 different jobs that turned out to be faulty ones, while the bottom two rows corresponds to jobs that were not faulty.

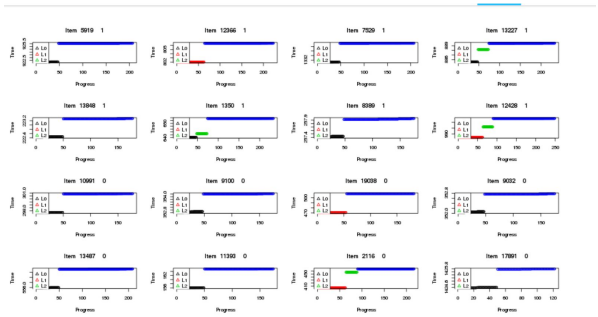


*Figure 6

*Figure: Progress of a job with respect to time

We also found out that the stations visited by faulty and non faulty nodes varies. There are some stations that are more frequented by faulty jobs, while there are some stations that are more frequented by the non faulty jobs.

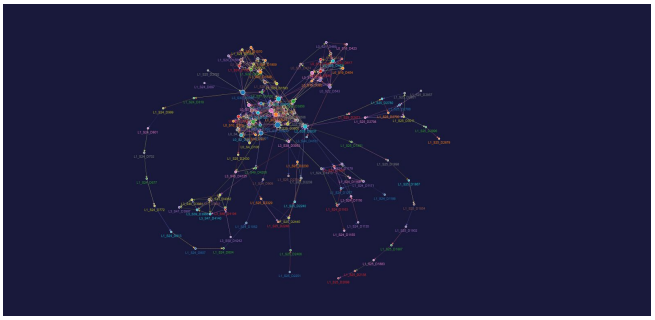


Figure 7

Figure: Graph depicting the movements of First 100 Faulty products from one station to another

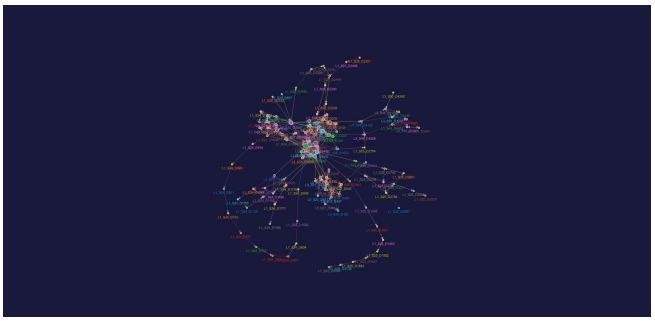


Figure 8

Figure: Graph depicting the movements of First 100 Non - Faulty products from one station to another

B. Feature Engineering

Building upon the insights gained in our initial explorations, we could interpret that there is a smaller subset of features that has a greater importance than the rest. Since the set of possible features is very large, we try different methods for dimensionality reduction. We used the domain knowledge of the data to create features that were later used in the classification procedure. Feature engineering is a crucial step in our workflow because of the huge data size and interesting inferences about their relative importance in deciding the failure of a product.

Drawing upon the conclusions of previous work done on this specific problem, we used Extreme Gradient Boosting[11] (XGBoost) to extract the top 15 useful features that dominated the results of any particular model. XGBoost is an advanced algorithm built on Gradient boosting.

*please find larger and clearer images in Appendix

We used XGBoost as it is faster and it implements parallel processing. It's implementation on Hadoop.XGBoost is a regularised algorithm, thus making the chances of overfitting lesser.
We developed our classification algorithms over these sets of useful features.

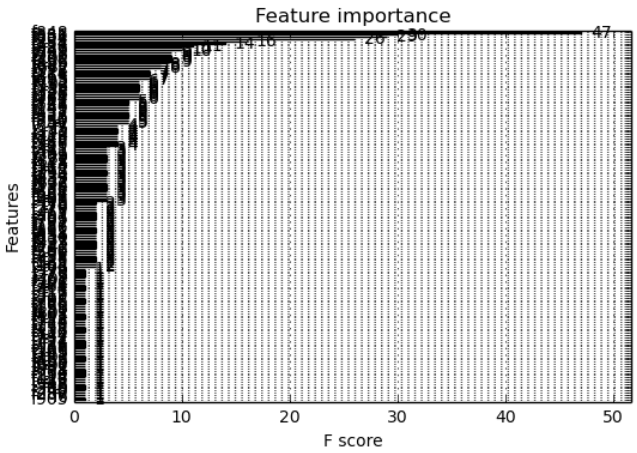


Figure 9

Figure: Relative Importance of Features

The above plot, depicts the relative importance of various features. The top 15 features that we found out on the basis of our analysis are the following: ('L3_S31_F3846', 'L1_S24_F1578', 'L3_S33_F3857', 'L1_S24_F1406', 'L3_S29_F3348', 'L3_S33_F3863', 'L3_S29_F3427', 'L3_S37_F3950', 'L0_S9_F170', 'L3_S29_F3321', 'L1_S24_F1346', 'L3_S32_F3850', 'L3_S30_F3514', 'L1_S24_F1366', 'L2_S26_F3036'). In the top 15 features we have 4 features corresponding to station 24, 3 features for station 33. We also observe that features corresponding to Line 1 and Line 3 are the most important ones. These observations are very much in accordance with our earlier data exploratory analysis.

C. Classification Algorithms

After having engineered the relevant features, we use the conventional classification techniques from machine learning namely: Naive Bayes Classifier, Multi-Layer perceptron, Random Forests [3] and Gradient Boosting (exponential) [4]. Since we have a very large amount of data (almost 7 GB of training data) and also a large number of features, only those algorithms that can work on chunks of data at a time, or/and in a distributive manner could be used effectively. Random Forest and Gradient boosting seemed to be the most suited algorithms for the classification task at hand (we also had prior knowledge that these two models give the best results, and their performance is comparable).

1. Random Forest [3] : An ensemble learning method constructed using a multitude of decision trees at training time. Each tree votes for a class and

response is assigned to data point based on majority voting. Ensembles are a divide-and-conquer approach used to improve performance. The main principle behind ensemble methods is that a group of "weak learners" can come together to form a "strong learner". Each classifier, individually, is a "weak learner," while all the classifiers taken together are a "strong learner".

2. Gradient Boosting [4]: Gradient boosting is also an ensemble of weak prediction models, typically decision trees that employs both gradient descent and boosting. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

As for the tools, we have used Spark, Python and R for formulating the model design and running out experiments and for visualisation and inference purposes, we have used IBM System G Tools and Databricks platform. All these tools are open source and we'll present the general description of the most relevant ones, namely Spark, System G and Databricks in the next section.

VI. Software Package Description

1. Python Scipy [5]: SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.
2. Python Numpy [6]: NumPy is the fundamental package for scientific computing with Python. It contains among other things:
    - a powerful N-dimensional array object
    - sophisticated (broadcasting) functions
    - tools for integrating C/C++ and Fortran code
    - useful linear algebra, Fourier transform, and random number capabilities

    Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

3. Python Pandas [7]: pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Some important Library Highlights:
    - A fast and efficient DataFrame object for data manipulation with integrated indexing;
    - Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
    - Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
    - Flexible reshaping and pivoting of data sets;
    - Intelligent label-based slicing, fancy indexing, and subsetting of large data sets;
    - Columns can be inserted and deleted from data structures for size mutability;
    - Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets;
    - High performance merging and joining of data sets;
    - Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
    - Time series-functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
    - Highly optimized for performance, with critical code paths written in Cythonor C.
    - Python with *pandas* is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.
4. Databricks[8]:Databricks' vision is to empower anyone to easily build and deploy advanced analytics solutions, a powerful open source data processing engine built for sophisticated analytics, ease of use, and speed. Databricks is the largest contributor to the open source Apache Spark project providing 10x more code than any other company. Databricks provides a just-in-time data platform, to simplify data integration, real-time

*please find larger and clearer images in Appendix

experimentation, and robust deployment of production applications.

5. IBM System G[9]: IBM System G is a comprehensive set of Graph Computing Tools, Cloud, and Solutions for Big Data. "G" stands for graphs -- large or small, static or dynamic, topological or semantic, and property or bayesian

VII. Experiment Results

We report a 10 Fold Cross Validation MCC scores for each of the classification algorithms, namely: Random Forests, Gradient Boosting, LibSVM, SVM with RBF Kernel, Gaussian Naive Bayes. Features for these classification algorithms were chosen after the result of the feature engineering test. The following table tabulates our results over the chosen set of features and classifiers.

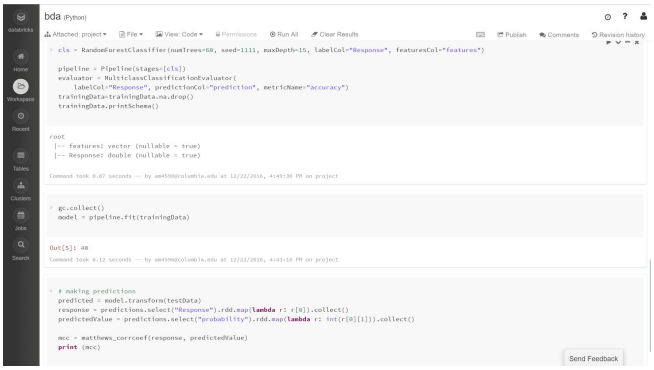


Figure 10
Figure: Snapshots of running spark scripts on databricks platform

Classification Algorithm	MCC Score (10 Fold Cross Validation)
Random Forests	0.39768
Gradient Boosting	0.414255
LibSVM	-8.37e-05
SVM with RBF Kernel	0.03225
Gaussian Naive Bayes	0.03370

Table: 10 Fold Cross Validation MCC Scores of Different Classification Algorithms

*please find larger and clearer images in Appendix

VIII. Conclusion

Member	Aayush Mudgal [am4590]	Sheallika Singh [ss5136]	Vibhuti Mahajan [vm2486]
Contribution (Fraction)	1/3	1/3	1/3

Table: Individual Member's Contribution

1. Gradient Boosting and Random Forests give the best and comparable results (largest MCC). Also these methods are more suitable for working on a big dataset because of relative ease with which the algorithm can be broken into independent chunks to be queued into a big data platform.
2. The probability of finding a defect after just seeing a defect was calculated as 10.08% as compared to the probability of finding a defect which is 0.52%. It is thus more probable to find a defect, just after a defect. This implies that the defected parts comes in batches.
3. Some stations and lines have more importance in deciding, suggesting the concept of backup lines, and special lines to detect defects more closely. This is clear from the system G visualisation (More input channels into S_24 and S_25). During feature selection we could give a concrete proof to our beliefs as these stations were the main features extracted while feature engineering

In the current implementation, we have ignored the duplicate values completely, however we would like to explore more about the importance of these values, and develop a suitable ranking algorithm for combining them.

Acknowledgment

We would like to thank professor Ching Yung Lin for his indelible teachings and all the TA's of EECS6893 (fall 2016) for their constant guidance and support in coursework, assignments and project.

References

[1] J. J. Bartko, “The intraclass correlation coefficient as a measure of reliability,” *Psychological Reports*, vol. 19, pp. 3–11, 1965.

[2] D. M. W. Powers, “Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation,” *J. Mach. Learn. Technol.*, vol. 2, pp. 37–63, 2010.

[3] T. K. Ho, “Random Decision Forests,” in *Intl. Conf. on Document Analysis and Recognition (ICDAR)*, 1994, vol. 1.

[4] F. Jerome H, “Stochastic gradient boosting,” *Comput. Stat. & Data Anal.*, vol. 38, pp. 367–378, Jan. 2002.

[5] E. Jones, T. Oliphant, and P. Peterson, “SciPy: Open source scientific tools for Python,” *http://www.scipy.org/*, 2000.

[6] T. Lindblad and J. M. Kinser, “NumPy, SciPy and Python Image Library,” in *Image Processing using Pulse-Coupled Neural Networks*, Springer

[7] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. “ O’Reilly Media, Inc.,” 2011.

[8] https://databricks.com/

[9] http://systemg.research.ibm.com/

[10] Warren Brussee, “Six SIGMA on a Budget: Achieving More with Less Using the Principles of Six SIGMA”

[11] Friedman et al., “Greedy function Approximation: A gradient Boosting Machine”, IMS 1999

[12] Link to Kaggle Challenge: https://www.kaggle.com/c/bosch-production-line-performance

[13] Link to Kaggle Forums: https://www.kaggle.com/c/bosch-production-line-performance/forums

*please find larger and clearer images in Appendix

Appendix [Enlarged Images]

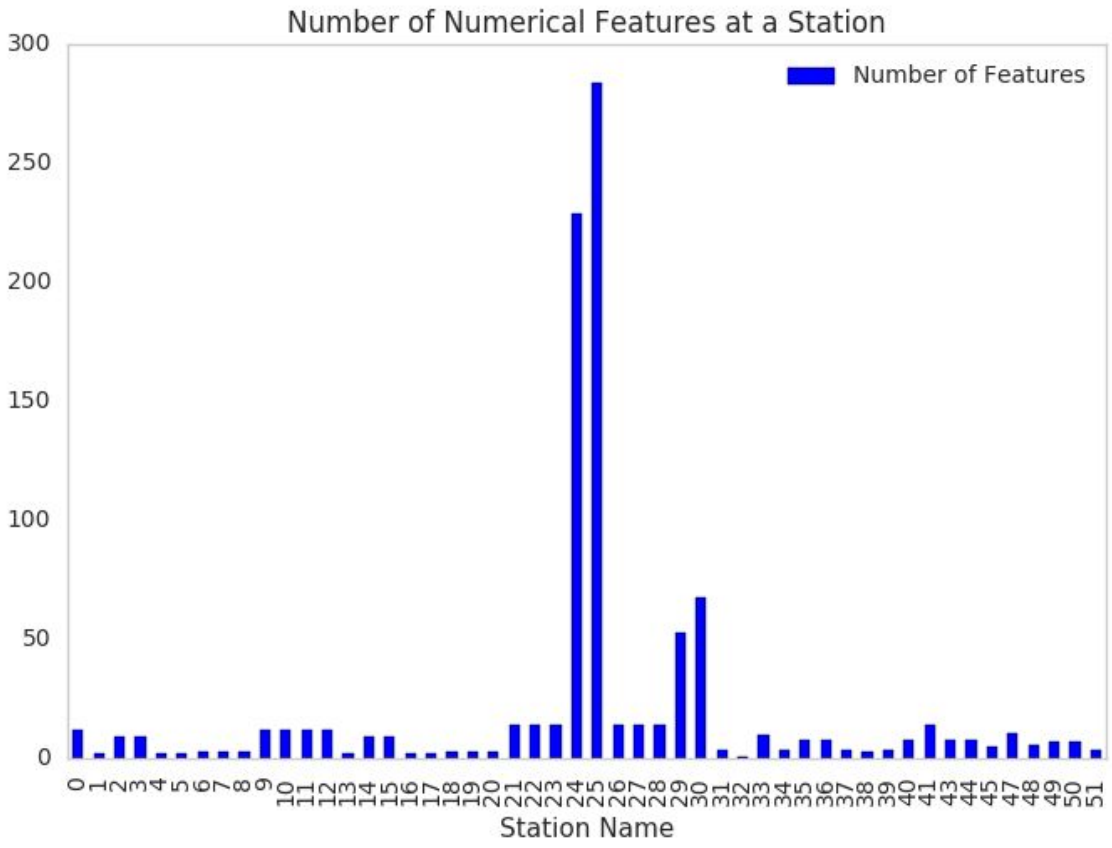


Figure 1

*please find larger and clearer images in Appendix

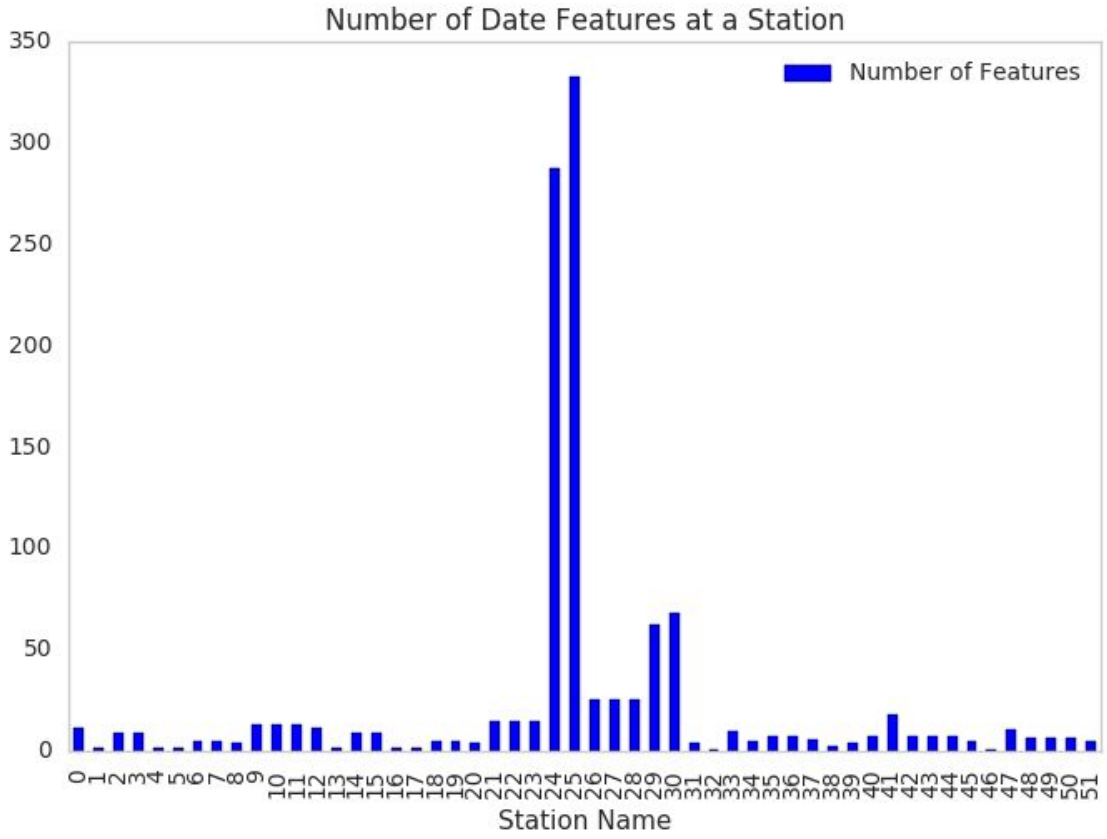


Figure 2

*please find larger and clearer images in Appendix

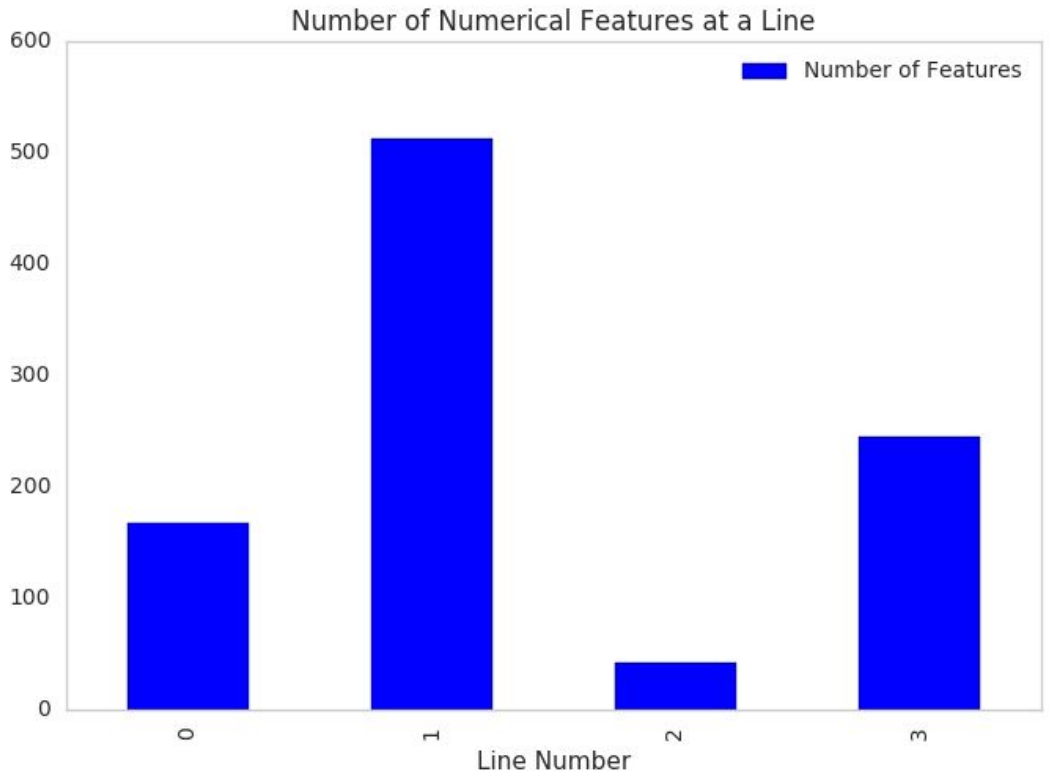


Figure 3

*please find larger and clearer images in Appendix

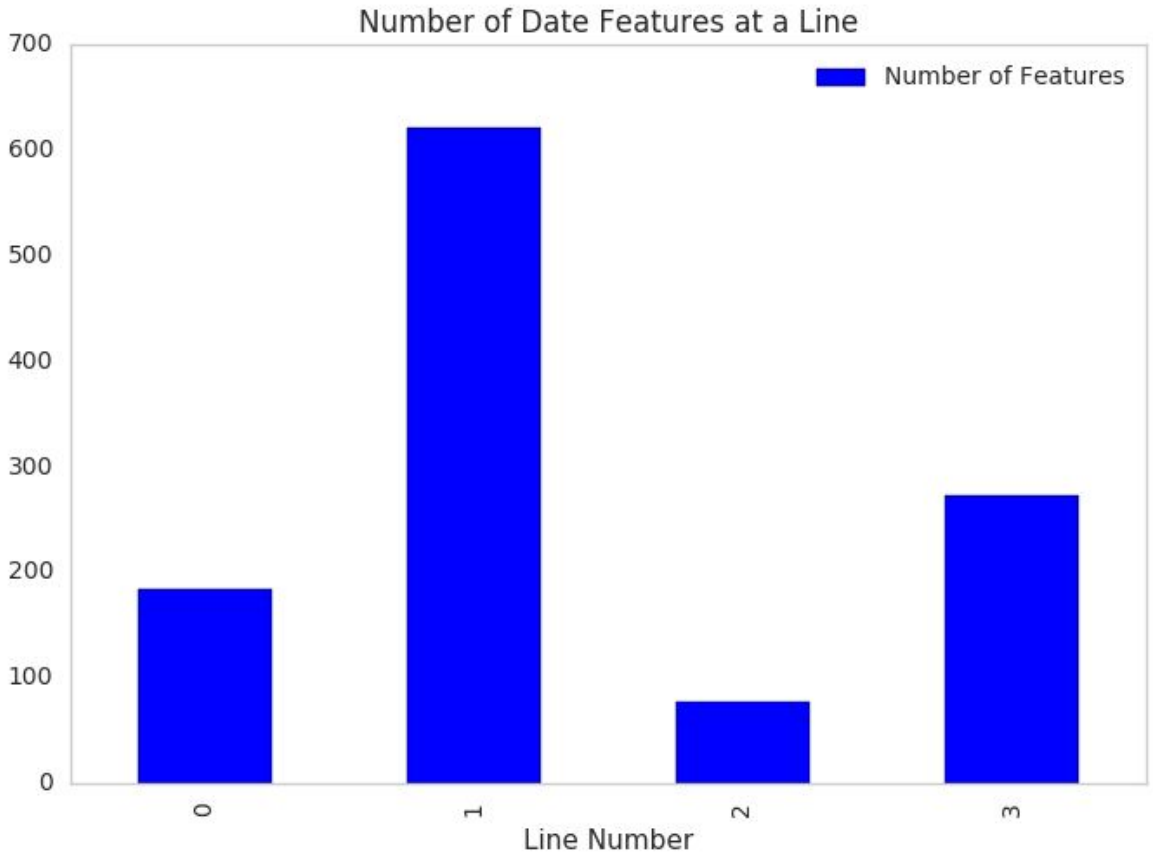
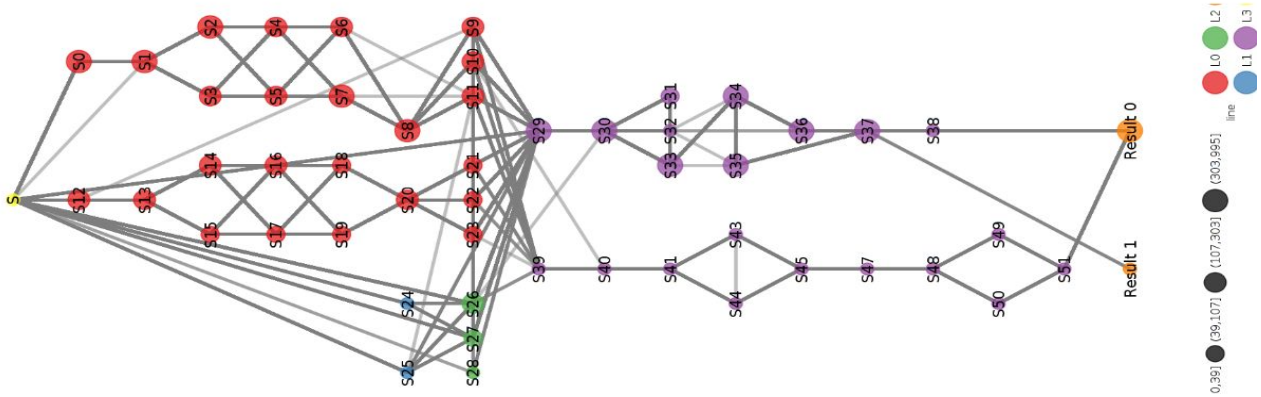
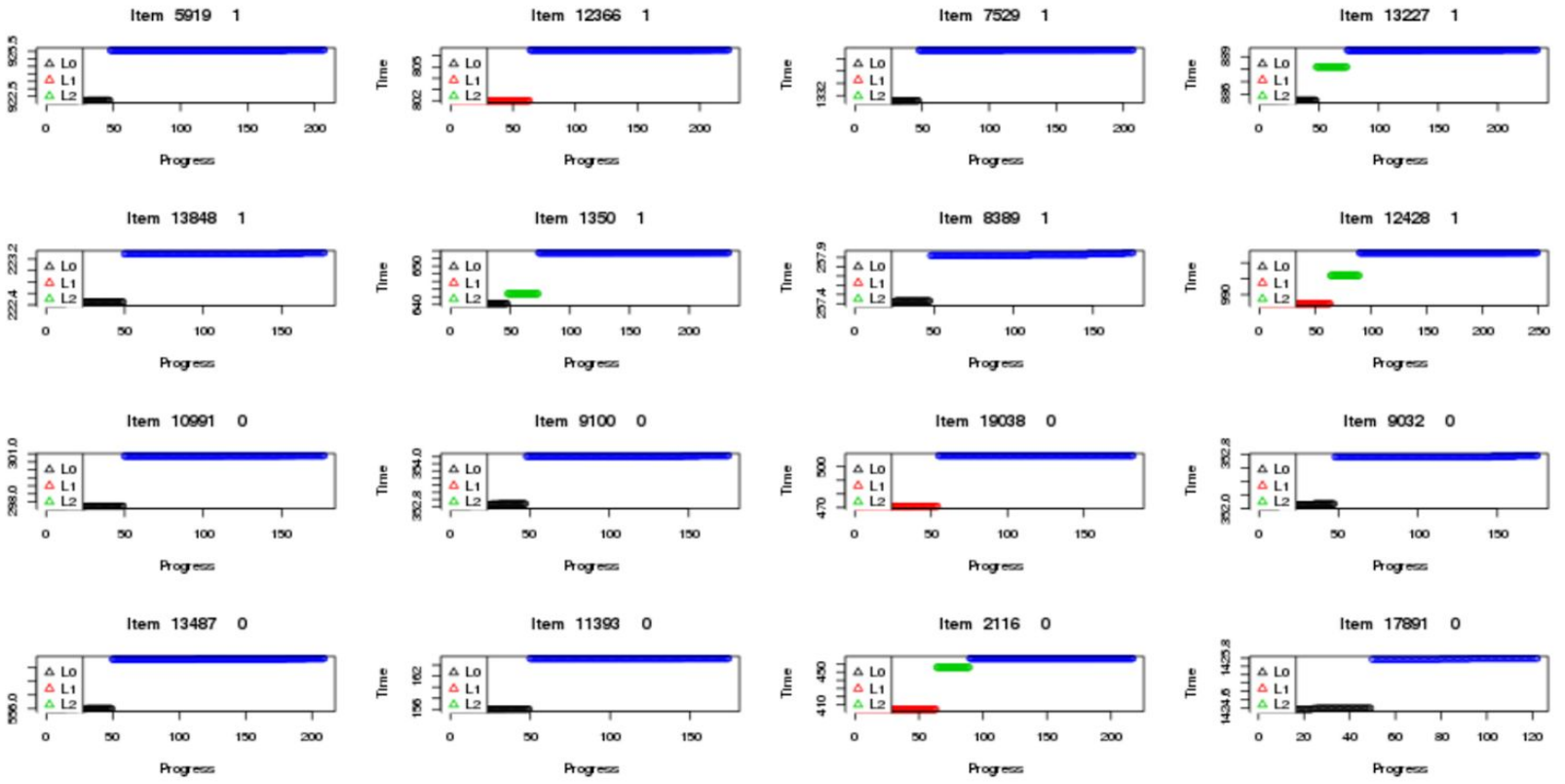


Figure 4

Figure 5: A graphical representation of the first 1000 jobs



*please find larger and clearer images in Appendix



Help sought from: [Kaggle Forums - JeffH]

Figure 6: Progress of a job with respect to time

*please find larger and clearer images in Appendix

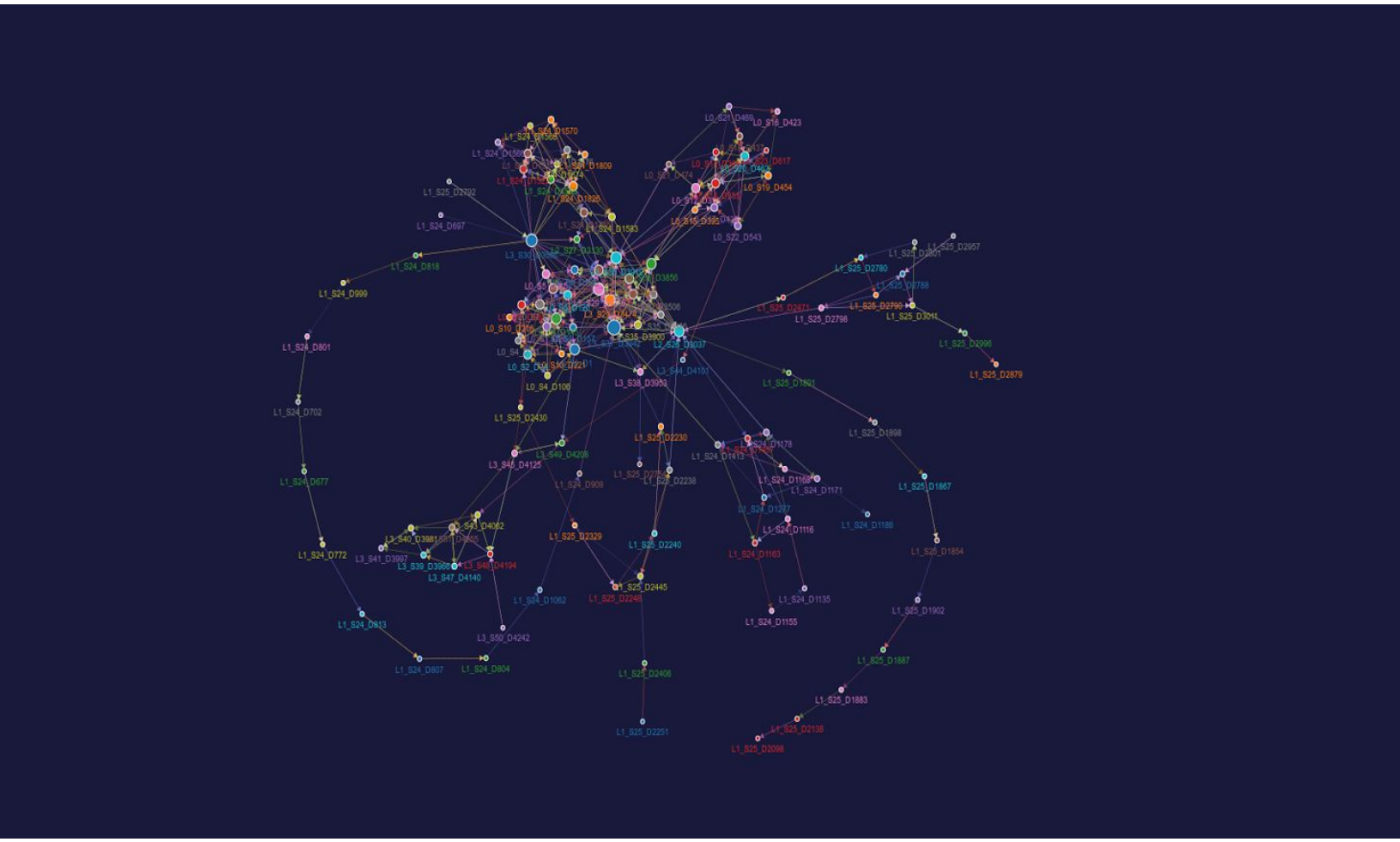


Figure 7 : Graph depicting the movements of First 100 Faulty products from one station to another

*please find larger and clearer images in Appendix

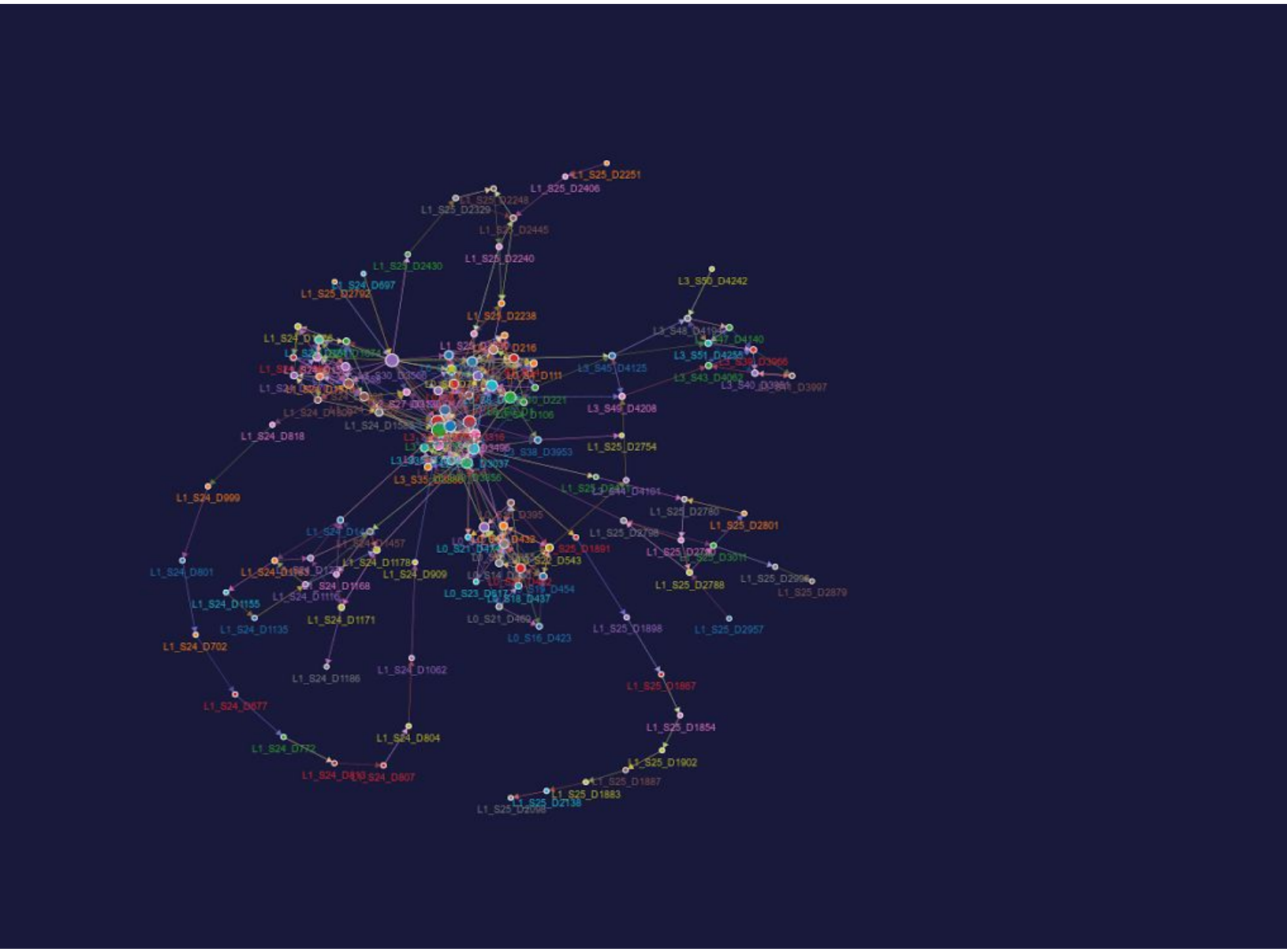


Figure 8: Graph depicting the movements of First 100 Non - Faulty products from one station to another

*please find larger and clearer images in Appendix

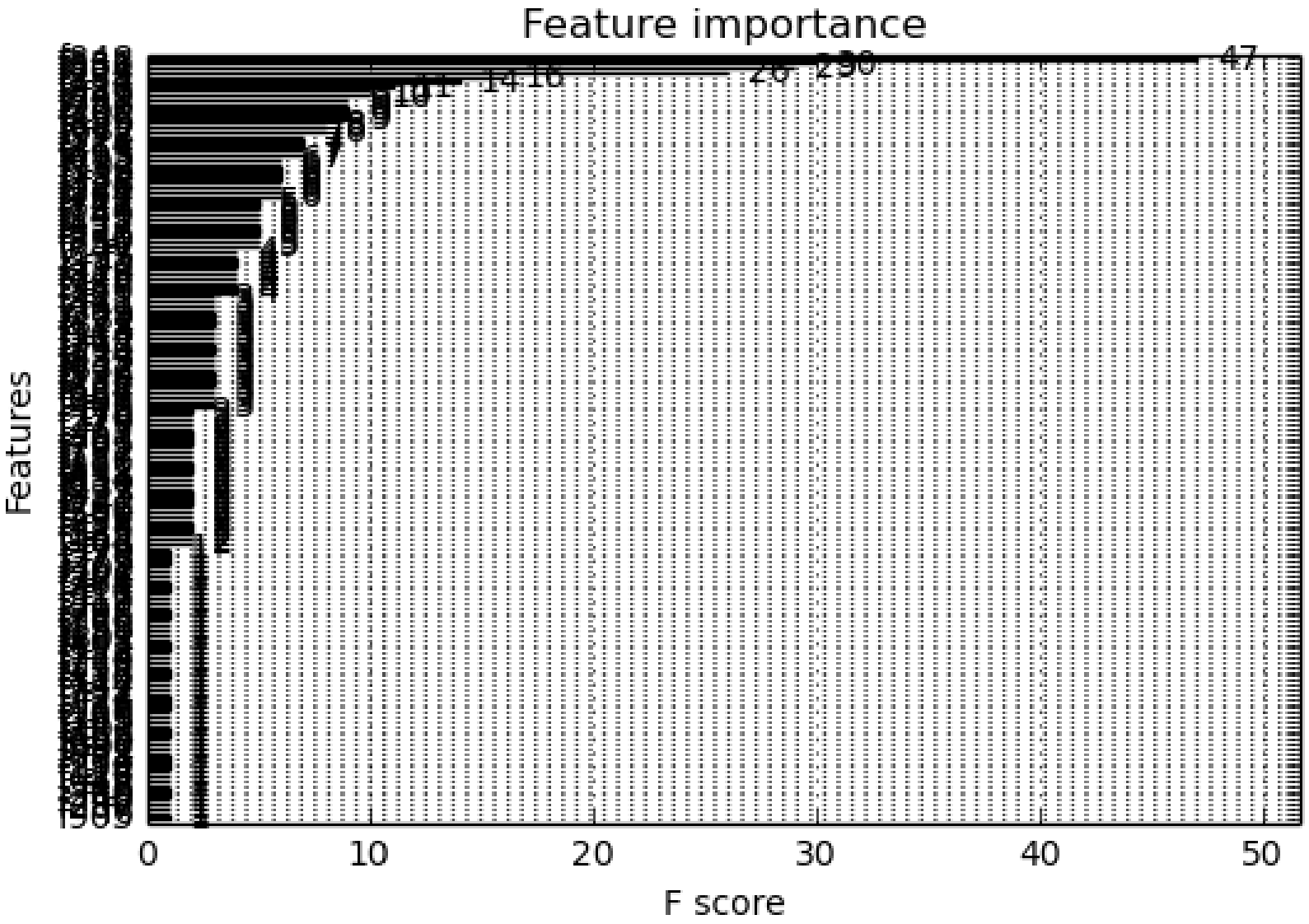


Figure 9

*please find larger and clearer images in Appendix

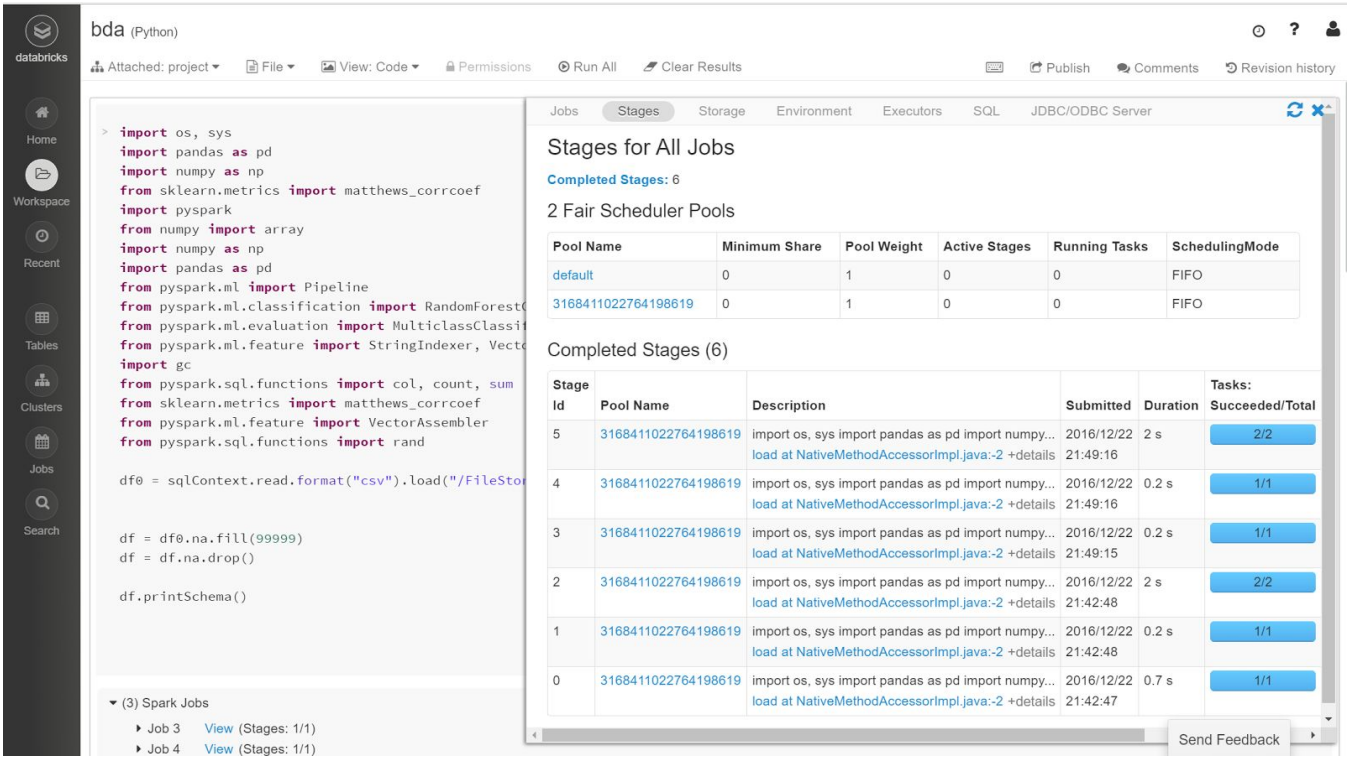
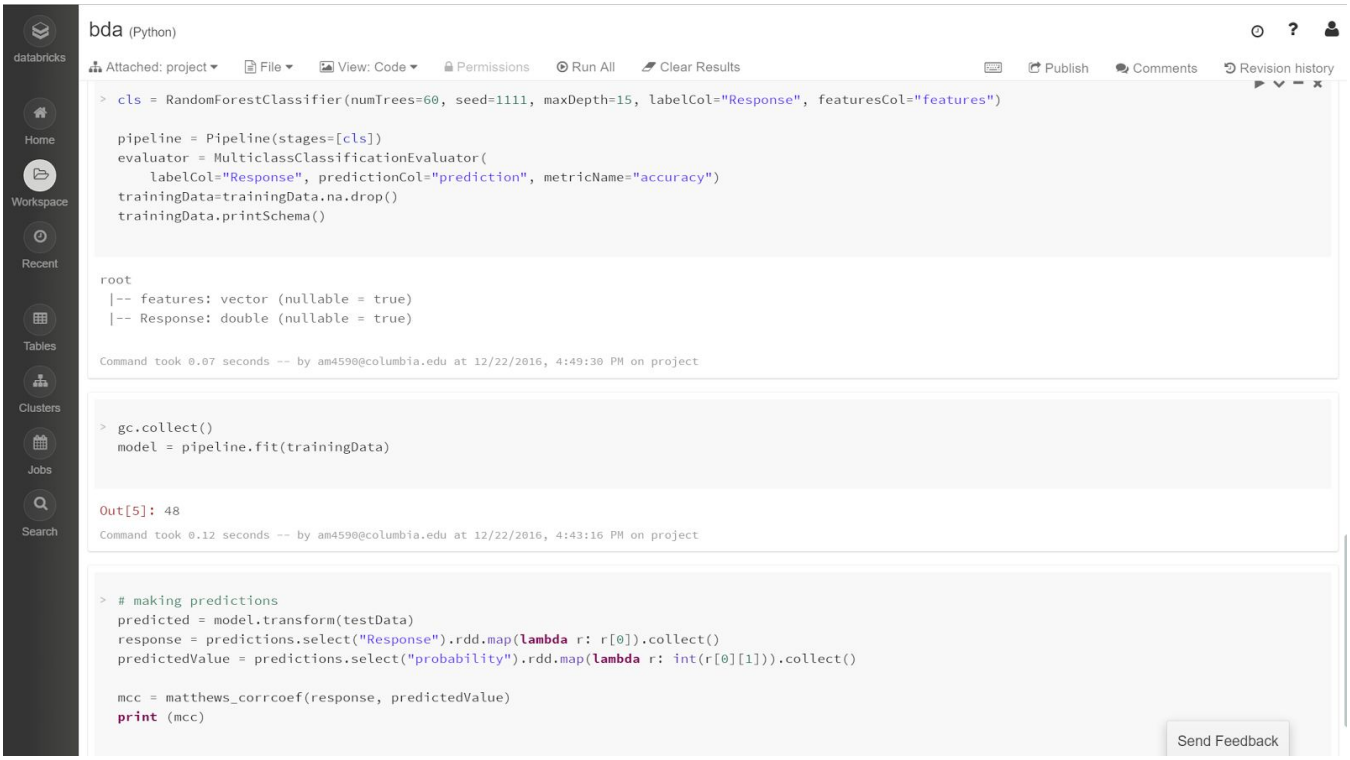


Figure 10



Snapshots of Running PySpark scripts on Databricks Platform
Figure 11

*please find larger and clearer images in Appendix