Practicum Final Report

Steelcase Answer Components Value Stream, Athens Plant

Prepared By: Maryam Rahmanpour
Krishna Kumar Ganesh
Chhavi Kadian

Sponsor: Steelcase Inc.

EXECUTIVE SUMMARY

Steelcase is a global, industry leading manufacturer of furniture, architecture, and technology products for workplace. This project report will walk through some of the key areas of focus in the overarching goal of building predictive models to improve the quality performance. The Answer value stream in specific will be covered in greater depth to build insights and provide the overall executive strategy to improve on the defects. Many strategies were used in data exploration to find out key insights about the defect types, claim types, product lines, areas responsible etc. While it was an initial challenge to merge the various data files together for further data cleaning and analysis, after all the methodology implementation, the data was ready for model creation, the results were quite impressive as can be seen in the report below. The two key features that were the results from the models are 'Volume of Orders' and 'Work Centers – other' for the 3 major defect codes – 'Missing', 'Wrong Style', and 'Wrong Finish'. Some of the recommendations for the overall execution strategy are making sure the Warehouse Management System is in place, application of lean principles to reduce waste at these feature levels, using barcode scanners to reduce human error, and six sigma principles to reduce variation in the process.

Table of Contents

1. INTRODUCTION

1.1. PROBLEM STATEMENT

As Steelcase wants to expand its current prediction-based performance improvement efforts in North America operations, the company wants to explore more failures and product lines to build a more proactive risk profile of sales order. The intent is to build a model which predicts the likelihood of certain quality defects in Answer Components value stream to help improve the overall quality performance.

1.2. OBJECTIVE

The main objective is to help scale prediction-based quality improvements to all parts of Operations. While improving the current model by adding more product lines and exploring more defect codes, the company is trying to improve its predicting ability. The focus will be to determine the relevant features for prediction and build predictive models that improves the quality performance proactively.

1.3. EXPECTED OUTCOME

The expected outcome of this project is to focus on key areas to achieve the following.

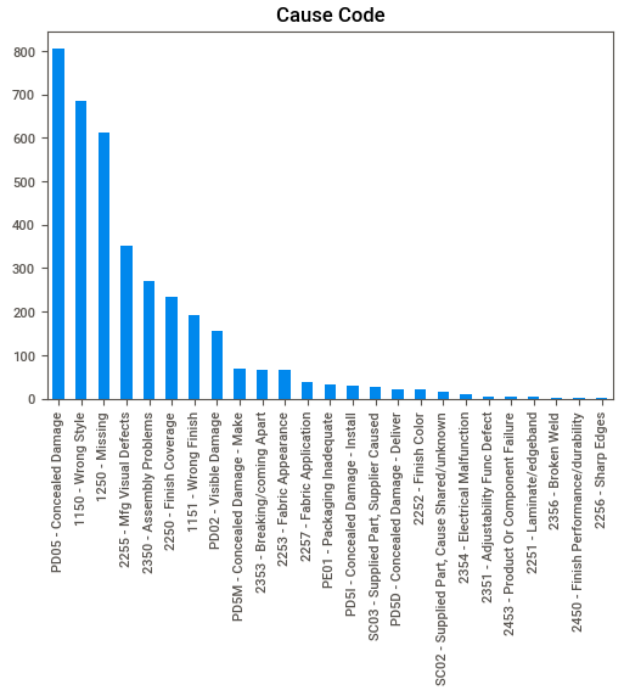
- Identify which manufacturing related defects to focus on
- Identify which product lines from this value stream can be targeted
- Determine factors that have a detrimental impact on quality
- Develop a predictive model to detect high risk orders
- Develop execution strategy for Operations
- Provide feedback on what can be changed in how we collect data to better facilitate data science projects

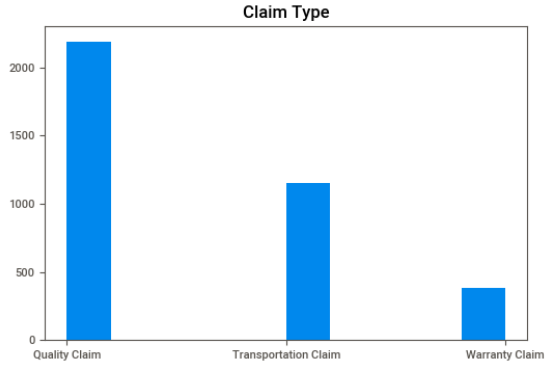
1.4. DATASETS PROVIDED

- Customer complaints data, Sales data, and Internal damage data
- Over 150 unique data points for each sales order plus claim pictures
- Structured, Imbalanced and Categorical

1.5. DATA EXPLORATION

Initially, data was explored to see some high-level common defects, causes of the defects, types of claims which were processed, which product line had more defects and finally, which areas of the production were identified to be the causes of defects.

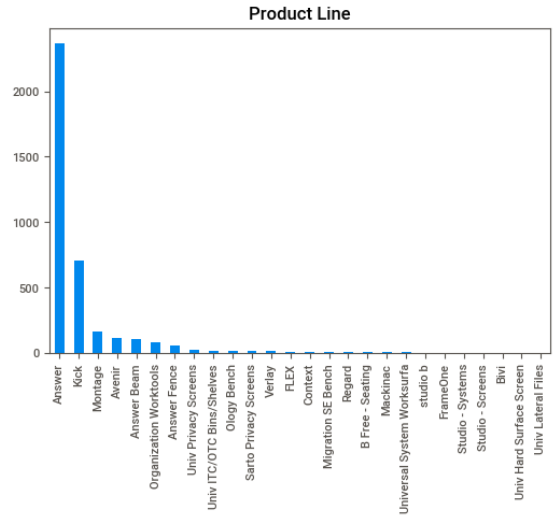


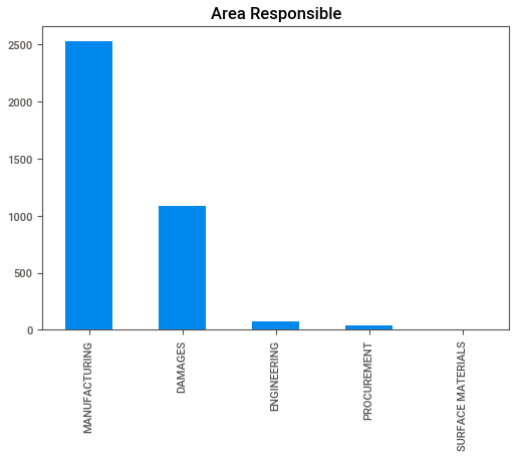


As can be seen from the Cause Code plot above, the top three contributing defect codes were:

1. Concealed Damage
2. Wrong Style
3. Missing

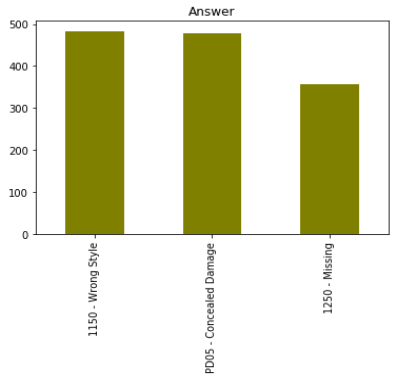
Quality claims seem to be the highest followed by transportation and warranty claims. Based on the objective of this project, this allows us to dig deeper into Quality claims as Steelcase group is interested in finding out how predictive modeling can assist them improving the overall quality of their products.

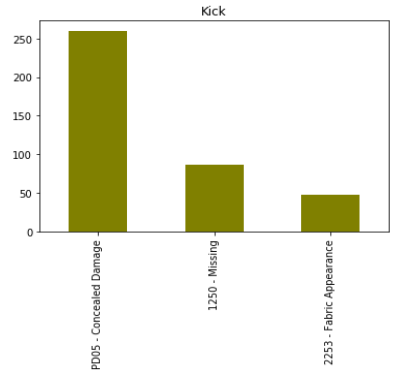


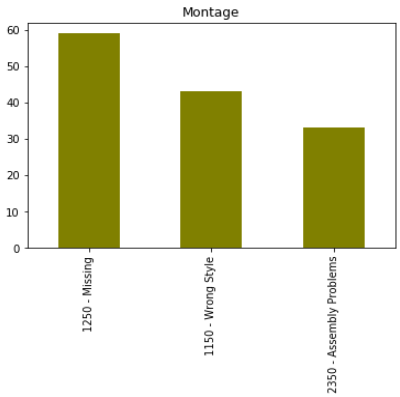


It is also important to note the results above were derived from the defects file. The top three product lines were identified to be Answer, Kick and Montage whereas the major areas contributing to the defects were recognized to be mainly from manufacturing and damages.

Also, the following charts show the product lines with most defects and for each product line and their associated top 3 cause code.



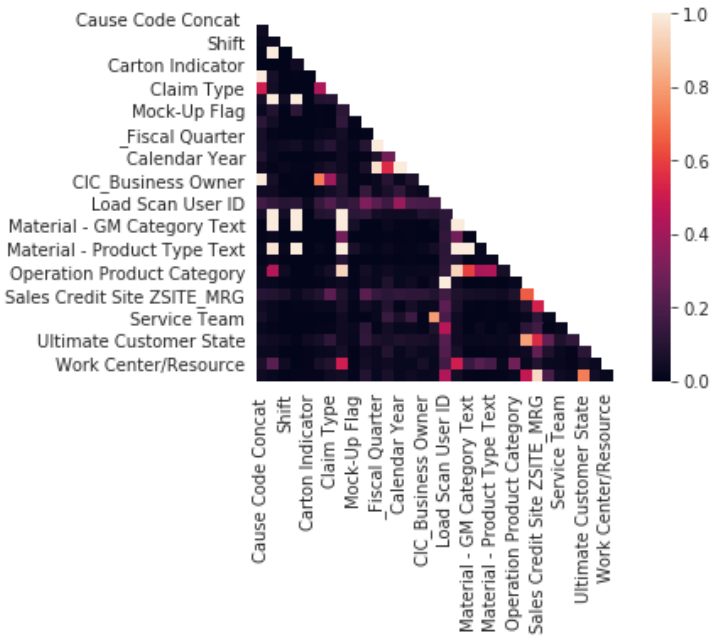




A word cloud was derived using customer feedback from a user input field on the claims form to identify any patterns that might emerge from the customer complaints. Certain word fillers were removed i.e., like, and, if, they, etc. to eliminate any possible noise.

From the word cloud above, we can clearly see the most mentioned keywords are missing, skin, panel, and trim.

Further to our data exploration, a heat map was also created to find out any correlation between the different features. As the heat map is mostly black, it was concluded that there is no strong correlation between the different features.



1.6. PROJECT APPROACH

In this project, the goal is to predict if an order would be suspected to claim or not. There is different information about each order coming in different datasets that needed to be merged. Data needs cleaning and some preparation steps. For example, there were some incorrect values for some features, the type of some columns should be unified in different datasets since we merge data on them, many columns should be dropped because of discussions during office hours and the opinion of experts in the company, etc. In most cases, machine learning is not done with the raw features. Features are transformed or combined to form new features in more predictive forms. This process is known as feature engineering. In many cases, good feature engineering is more important than the details of the machine learning model used. Good features can often make even poor machine learning models work well, whereas, given poor features, even the best machine learning model will produce poor results. As the famous saying goes, "garbage in, garbage out".

2. METHODOLOGY 1 – DOWNSAMPLING BASED MODELING

2.1. EXTENDED DATA ANALYSIS

Some common approaches to feature engineering include:

Aggregating categories of categorical variables to reduce the number. Categorical features or labels with too many unique categories will limit the predictive power of a machine learning model. Aggregating categories can improve this situation, sometimes greatly. However, one must be careful. It only makes sense to aggregate categories that are similar in the domain of the problem. Thus, domain expertise must be applied.

2.2. FEATURE ENGINEERING

Feature engineering uses domain knowledge of the data to create features that make machine learning algorithms work. It is the act of extracting important features from raw data and transforming them into formats that is suitable for machine learning. (Wikipedia)

Therefore, in this experiment, we performed feature engineering (aggregating different categories in each feature) with two primary goals as follows:

1. Preparing the proper input dataset
2. Improving the performance of machine learning models

2.3. DATA PREPARATION FOR MODELING

First, we applied ML models to data containing all defect causes. In the second round as it was asked by Steelcase experts, we only focused on 3 defect causes of the claims. Defect causes are: “Missing”, “Wrong style”, and “Wrong finish”. So, the data is filtered based on these three cause codes. The AUC score dropped in the second round. The reason could be that other defects cause do not make noise in data and they contain valuable information, and in the best-case filtering data by defect cause is a subset of training data and the AUC of the model can be equal to or smaller than that of the trained model on data with all-cause codes.

2.3.1. CLASS BALANCE

An imbalanced classification problem is an example of a classification problem where the distribution of examples across the known classes is biased or skewed. The distribution can vary from a slight bias to a severe imbalance. Imbalanced classifications pose a challenge for predictive modeling as most of the machine learning algorithms used for classification were designed around the assumption of an equal number of examples for each class. This results in models that have poor predictive performance, specifically for the minority class. This is a problem because typically, the minority class is more important, and therefore the problem is more sensitive to classification errors for the minority class than the majority class. [A Gentle Introduction to Imbalanced Classification]

In this project, Data is strictly unbalanced, and the ratio of claims is 0.24 % of all sales data. To deal with this problem, two different approaches have been employed. Downsampling and Upsampling.

2.3.2. DOWNSAMPLING APPROACH

Downsampling is a mechanism that reduces the count of training samples falling under the majority class. As it helps to even up the counts of target categories. The caveat of downsampling is that we tend to lose so much valuable information. After downsampling, we keep about 3500 rows including orders with the claim and without claim.

After downsampling, we need to extract important features. Feature Selection is the process where we automatically or manually select those features which contribute most to our prediction variable or output in which we are interested. Having irrelevant features in our data can decrease the accuracy of the models and make our model learn based on irrelevant features. Three different feature selection approach has been considered in this study: “L1-based linear SVM feature selection”, “Lasso-based

feature selection", and "tree-based feature selection". [Feature Selection Techniques in Machine Learning with Python]

2.4. MODELING AND OUTPUT EVALUATION

2.4.1. L1-BASED LINEAR SVM FEATURE SELECTIO

Linear models penalized with the L1 norm have sparse solutions: many of their estimated coefficients are zero. When the goal is to reduce the dimensionality of the data to use with another classifier, they can be used along with SelectFromModel to select the non-zero coefficients. Sparse estimators useful for this purpose are the Lasso for regression. [Feature Selection]

The output of this feature selection method is:

Th output for data with all defects cause is:

```
['4ZSD_AD06_KBMENG', '4ZSD_AD06_VOLUM', '4ZSD_AD06_ZZLINEVOLUME',
 '4ZSD_AD06_ZZLINEWEIGHT', 'Pieces Shipped', '4ZSD_AD06_MATKL_T01',
 '4ZSD_AD06_VSTEL_EP00', '4ZSD_AD06_VSTEL_NT00',
 '4ZSD_AD06_ZZCONPATH_AA2EP', '4ZSD_AD06_ZZCONPATH_AA2GM',
 'PCC_ANSWER PANEL FRAMES', 'Product Line_Answer', 'Product Line_other',
 'Op Prod Category_other', 'Work Center_ANSWER CARTON TRIM',
 'Work Center_other']
```

And the result for data with only 3 major defect cause is:

```
['4ZSD_AD06_KBMENG', '4ZSD_AD06_VOLUM', '4ZSD_AD06_ZZLINEVOLUME',
 'Pieces Shipped', 'PCC_ANSWER PANEL FRAMES', 'PCC_other',
 'Product Line_Answer', 'Op Prod Category_other',
 'Work Center_BLACK LINE (VAR - PREVIOUSLY RFC)', 'Work Center_other']
```

2.4.2. LASSO BASED

Lasso involves a penalty factor that determines how many features are retained. When features are correlated, Lasso will choose one or the other based on its performance in the data sample at hand. [Feature Selection]

The output of Lasso based feature selection is as follows.

Th output for data with all defects cause is:

```
['4ZSD_AD06_KBMENG', '4ZSD_AD06_VOLUM', '4ZSD_AD06_ZZLINEVOLUME',
 '4ZSD_AD06_ZZLINEWEIGHT', 'PCC_other', 'Work Center_other']
```

And the result for data with only 3 major defect cause is:

```
['4ZSD_AD06_KBMENG', '4ZSD_AD06_VOLUM', 'Pieces Shipped', 'PCC_other',
 'Op Prod Category_other', 'Work Center_other']
```

2.4.3. TREE-BASED FEATURE SELECTION

Tree-based estimators can be used to compute impurity-based feature importance, which in turn can be used to discard irrelevant features, when coupled with the SelectFromModel meta-transformer. [Feature Selection]

The output for data with all defects cause is:

```
['4ZSD_AD06_BRGEW', '4ZSD_AD06_KBMENG', '4ZSD_AD06_NTGEW',
  '4ZSD_AD06_VOLUM', '4ZSD_AD06_ZZLINEVOLUME', '4ZSD_AD06_ZZLINEWEIGHT',
  'Pieces Shipped', 'Work Center_other']
```

And the result for data with only 3 major defect cause is:

```
['4ZSD_AD06_BRGEW', '4ZSD_AD06_KBMENG', '4ZSD_AD06_NTGEW',
 '4ZSD_AD06_VOLUM', '4ZSD_AD06_ZZLINEVOLUME', '4ZSD_AD06_ZZLINEWEIGHT',
 'Pieces Shipped']
```

2.5. MODEL SELECTION/SETUP AND HYPERPARAMETER TUNING

Scikit-Learn provides easy access to numerous different classification algorithms. Among these classifiers are:

1. Logistic Regression,
2. K-Nearest Neighbors,
3. Support Vector Machines,
4. Random Forest Naive Bayes XGBoost,
5. AdaBoost,
6. Deep Learning

At this point of analysis, although there might be some guesses regarding the candidate classification model for this specific dataset, we designed a pipeline through which all the mentioned models will be studied, and the best model would be selected using the AUC score.

It should be noted that Hyperparameter Tuning is also performed in this step. It's like a grid search that goes through all the combinations of different models and different hyperparameters, trains the model, predicts, and saves the AUC score in a data frame. Then the first model in the sorted data frame would be our best model with its specific hyperparameters.

Since for real-world applications, various scenarios may be defined, the time of training may be challenging too. Although cloud providers can provide more computational power, for this specific experiment 'run_parallel_computation' function is designed to use more local CPUs to faster the computations.

2.6.RESULTS AND RECOMMENDATIONS

The output of training different models with different hyperparameters is a data frame. The head of the data frame which shows the models is as follows.

Result for data with all defect causes:

	method_param_id	roc_auc_score	parameters	method_id	variable_selection_method
0	54	0.775665	{'n_estimators': 300,'max_depth': 4}	XGboost	l1
1	29	0.772228	{'batch_size': 64,'hidden_layer_sizes': 1}	DL	l1
2	52	0.771622	{'n_estimators': 300, 'max_depth': 4}	XGboost	tree
3	27	0.771015	{'n_estimators': 300, 'max_depth': 4}	XGboost	lasso
4	30	0.769423	{'batch_size': 32, 'hidden_layer_sizes': 3}	DL	l1

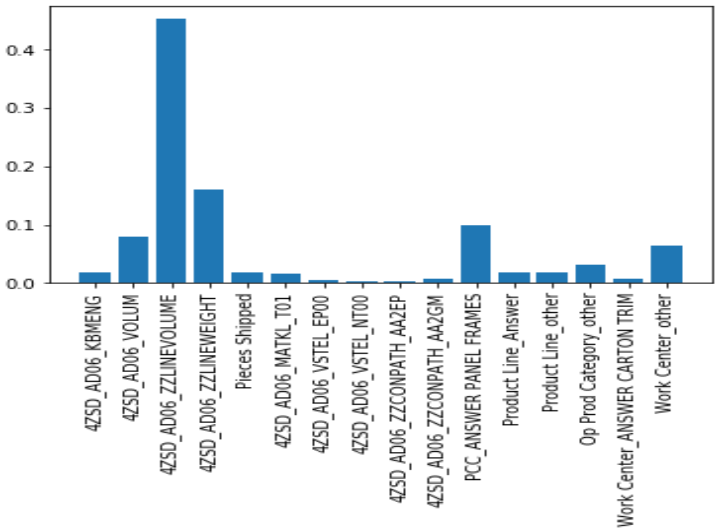
Results for data with 3 major defects cause:

	method_param_id	roc_auc_score	parameters	method_id	variable_selection_method
0	54	0.700197	{'n_estimators': 300, 'max_depth': 4}	XGboost	l1
1	29	0.697600	{'batch_size': 64, 'hidden_layer_sizes': 1}	DL	l1
2	52	0.694707	{'n_estimators': 150, 'max_depth': 4}	XGboost	l1
3	27	0.694159	{'batch_size': 32, 'hidden_layer_sizes': 3}	DL	l1
4	30	0.690630	{'batch_size': 64, 'hidden_layer_sizes': 2}	DL	l1

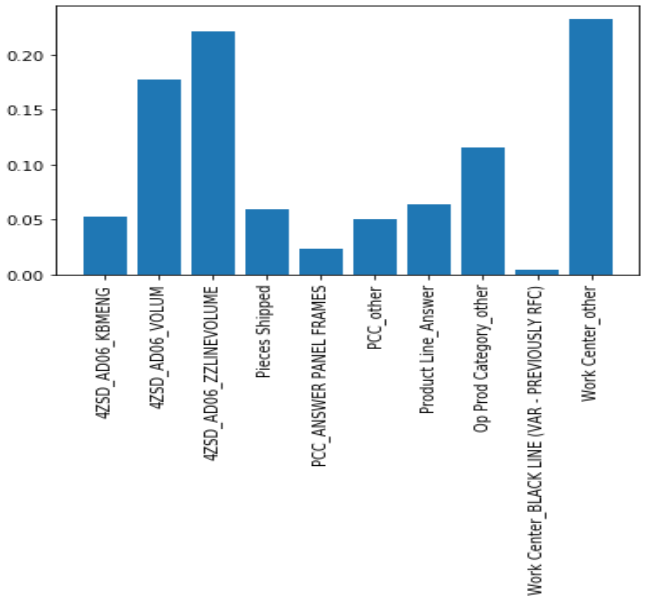
The results show that XGboost and DL models have the best AUC score. Since XGboost is a tree-based m odel, we can also see the feature importance of the model, As following image shows the Volume of the

orders and aggregation of work-center other than “ANSWER CARTON TRIM, BLACK LINE (VAR - PREVIOU SLY RFC), ANSWER FINAL, FRAMELESS GLASS” are the most important feature used in the model.

Important features of the Model with all defect cause:



Important features of the Model with 3 major defects cause:



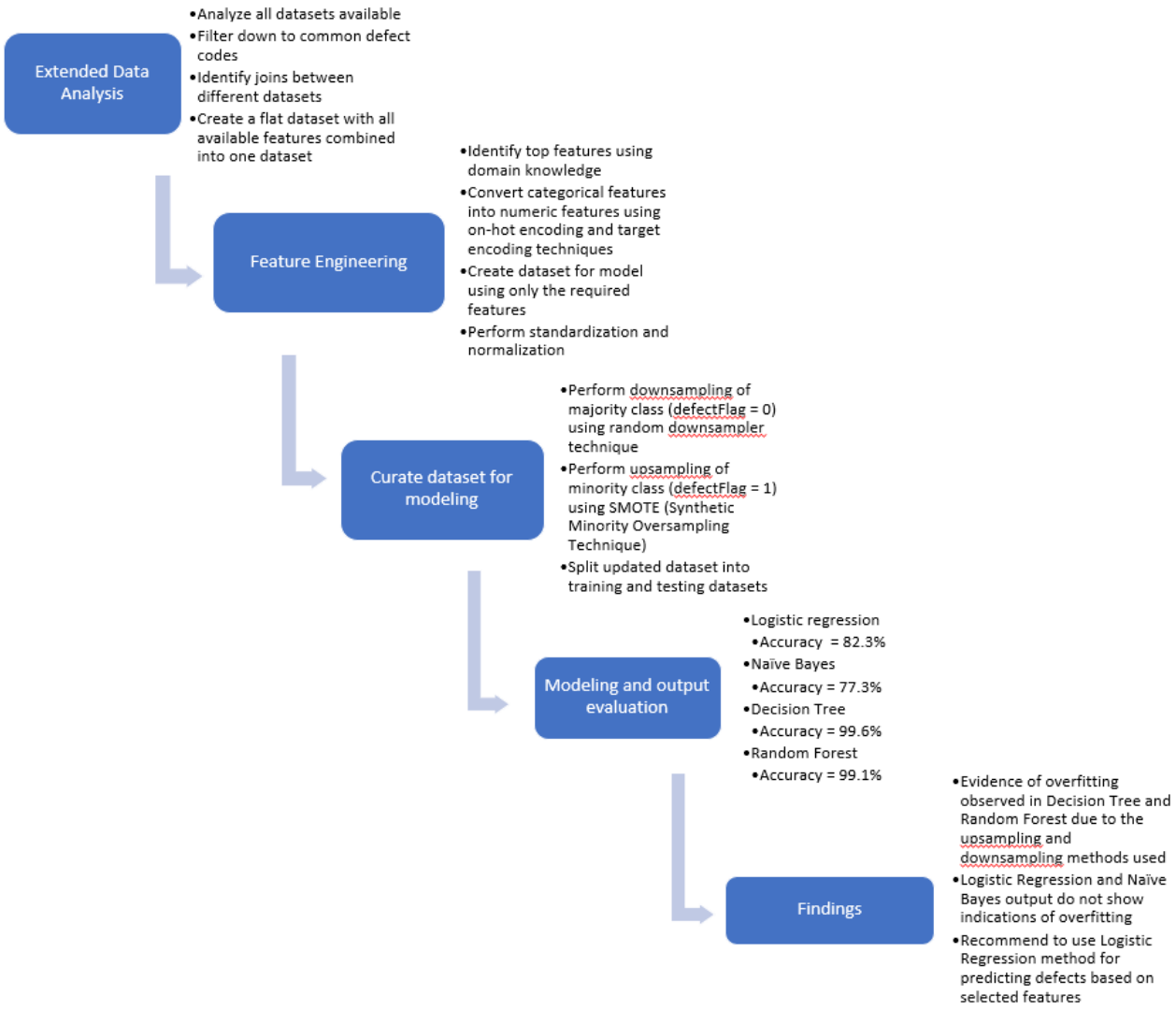
2.7.SOURCE CODE REPOSITORY

https://github.com/rrmaryam/Steelcase.git

3. METHODOLOGY 2 - HYBRID UPSAMPLING METHODOLOGY AND DOMAIN KNOWLEDGE BASED FEATURE ENGINEERING

Another approach we developed was to perform a hybrid of downsampling and upsampling of the output defect Flag and performing feature engineering using domain knowledge to identify the top features to use in the model. In summary, this approach identified another model which can potentially be used to predict orders which are at a high risk of ending up with a defect.

Below were the various steps in this approach:



3.1.EXTENDED DATA ANALYSIS

We analyzed the data points across the different datasets provided to find out the best way to merge all the different datasets together. Since the data had different data types across the files, some additional data cleanup was required to make the common fields between the different datasets in the same format so that we can successfully 'join' these datasets using the key columns.

3.2. FEATURE ENGINEERING

As part of the section above, we created a single output dataset which contained all relevant fields from the different datasets provided. Through analyzing the key fields in the input dataset, and understanding the purpose of the different fields, we were able to narrow down the column list on this consolidated dataset from over 100 features to just 24 (The features we narrowed down to can be reviewed/downloaded here). We also filtered the defects dataset to only orders which had one of the below 3 common error codes:

3.3. DATA PREPARATION FOR MODELING

The 24 features we identified were a mix of both categorical and numerical data types. Some of the categorical features had many different categories on them.

3.3.1. HANDLING CATEGORICAL DATA

We performed 2 different types of handling for the categorical columns depending on the number of distinct categories in the fields

- **One-Hot Encoding**

  This was performed for categorical columns which had 5 or less distinct values. This technique creates a set of new columns – one for each category in the file, populated with a 0 or 1 depending on whether the category value is present in the data or not.

- **Target Encoding**

  This technique was used for columns with large number of categories – The technique calculates the mean of the target variable for each category and replaces the category with the mean value.

3.3.2. Class Balance – Downsampling followed by upsampling

In this project, data was strictly unbalanced, and the ratio of claims is 0.24 % of all sales data. In this methodology, we used a hybrid approach – where we performed downsampling of the majority class first (defectFlag = 0), followed by an upsampling of the minority class (defectFlag = 1).

- **Downsampling**

The downsampling of the majority class was performed using the Random Undersampler. With this technique, we reduce the number of rows on the majority class by a factor we choose in the hyperparamters. While performing undersampling of the majority class in this methodology, we made sure that the dataset size does not reduce too significantly to avoid losing any valuable information from the data set. After downsampling, the dataset size reduced from around 700k rows to 150k rows.

- **Upsampling**

The upsampling was performed on top of the downsampled dataset, by upsampling the minority class. We used the SMOTE (Synthetic Minority Oversampling Technique) technique for oversampling the minority class. This technique analyzes the minority class records and tries to synthetically create new rows that have similar features as the minority class. This helps balance the number of rows in the minority and majority classes so that our model can be trained on more 'realistic' data. After upsampling, the total dataset size became approximately 250k rows.

3.3.3. Standardization and Normalization

Once the dataset was curated to have the classes in the right proportion, we then standardized and normalized the features to avoid features with larger values from influencing the model performance.

3.4. Modeling and Output Evaluation

3.4.1. Train and Test Datasets

We tried different splits of the input dataset for test and train datasets. After testing different combination of splits, a 75:25 split of the input dataset into testing and training datasets gave us the best performance.

3.4.2. MODEL COMPARISON

To validate this methodology, we ran multiple classification algorithm models using the scikit-learn package to evaluate the classification accuracy on the cured dataset. Below is how the top 4 models looked like:

Model Type	Parameters	Accuracy
Logistic Regression	Random state = 22, max_iter = 200	82.3%
Naïve Bayes	Priors=None, var_smoothing = 1e-09	77.38%
Decision Tree classifier	Splitter='best'	99.6%
Random Forest	n_estimators=100	99%

3.4.3. MODEL OUTPUT INFERENCES

Since the input dataset is curated using sampling methods which synthetically creates new rows based on input rows already on the dataset, and tree-based classification models like decision tree and random forest perform training and split the trees based on rows which are closely resembling each other, we believe the Decision Tree classifier and the Random Forest classifier models have had some overfitting on them which explains the very high accuracy on the model performance. We would need to test the model using newer data points to see if the model can perform similarly with unknown datasets.

However, the logistic regression model does not have this issue since it uses all the features on the input dataset in unison to perform model training. The accuracy on the model of 82.3% is also representative of its performance against the unknown test dataset. From this methodology, we would recommend using the Logistic Regression model for predicting if a defect can occur on the input dataset or not.

3.5.SOURCE CODE REPOSITORY

https://github.com/kk87736/georgiaTechCapstone

4. ASSUMPTIONS, ISSUES AND CONSTRAINTS

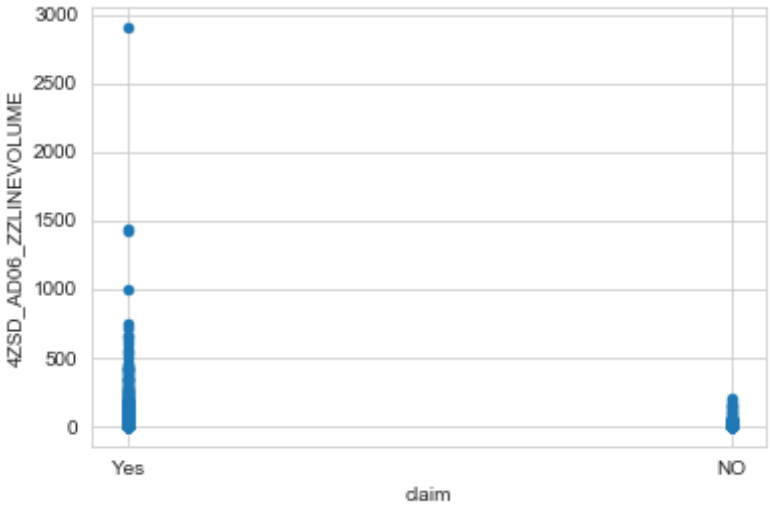
1. Categorical data was difficult to work with
2. Data collection was not unified. It was difficult to merge the data together.
3. Some data was stored in parcel level and some in the style level, which makes the merging difficult
4. Defect data logging/collection had errors and some attributes on the files had empty values

5. EXECUTION STRATEGY FOR OPERATIONS

On a high level, we must review the following 4 steps to assess where the organization is currently with the defect management, identify gaps and move closer to a zero-defect goal.

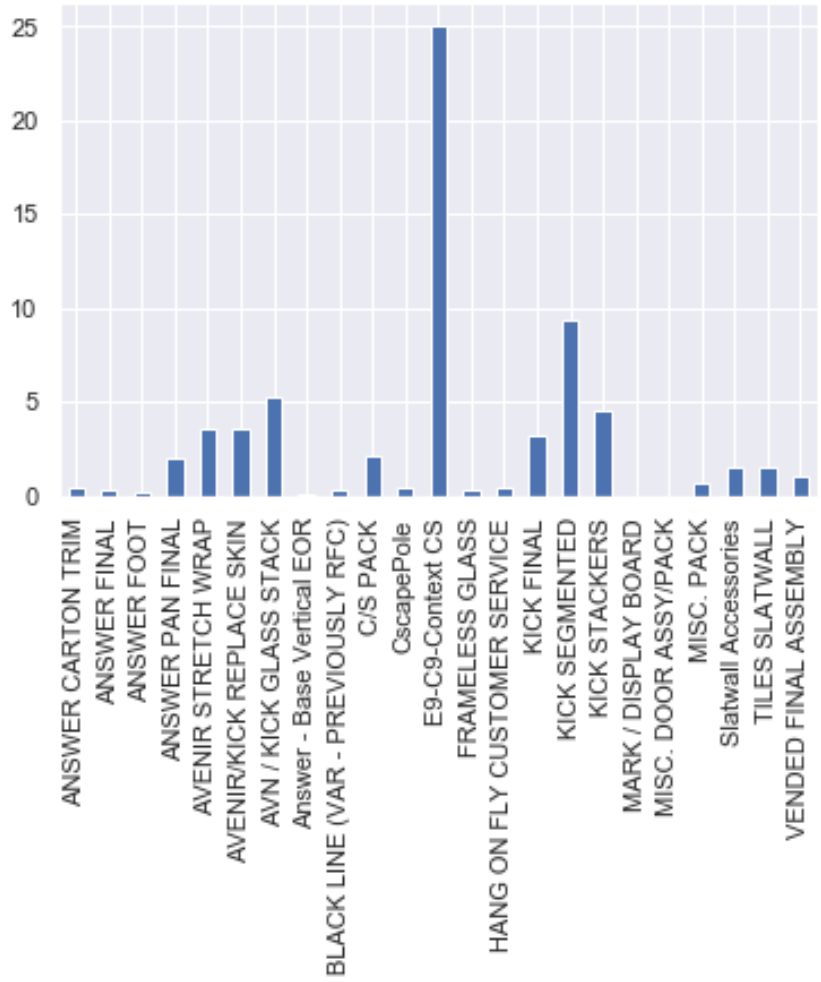
1. Defect Visibility and Defect Prioritization

From the models above, the main goal is to assess which order types are more prone to be claimed as defects. Even though the top defect codes were identified to be 'Concealed damage', 'wrong style', and 'missing'; Concealed damage was excluded as it is most likely happening after the product leaves the facility which is out of scope for this project. Therefore, the 3 defects which were studied for the purpose of this project were 'missing', 'wrong style', and 'wrong finish'. Using these 3 defect codes, 'Volume of the orders' seems to be the most important feature followed by 'Work Center – Other' where both play a substantial role in claiming an order with a defect. Since 'Volume of orders' feature is defined as the volume per order, we investigated the threshold volume above which the orders are more prone to be claimed as a defect.



As it can be seen from the plot above, where a test of 'Volume of Orders' was set to 200, it seems that number of orders claimed to be a defect far exceeded than not. It seems like larger orders with many parts are more susceptible to be having claimed with one of the 3 defect codes identified above.

The second most important feature being 'Work Centers – Other', we did a plot with all the categories which were grouped in this feature and found out that top contributing work center was 'E9-C9-Context CS' followed by 'Kick Segmented', 'AVN / Kick Glass Stack', and 'Kick Stackers'. We would need to further explore into these different work centers to find out the root causes of the defects caused at these centers.



2. Defect Resolution

Based on the two features which were identified by the models created in this project, we tied the 3 major defect codes together to come up with the overall strategy to optimize the quality performance.

Some of the overall improvement ideas focusing on these issues are (if not already in place):

1. Having a Warehouse Management System in place which can help increase accuracy with reduced cycle time.

2. Create Separate bins and use barcode scanning for the parts on each distinct order. This strategy will help prevent improper selection (wrong style and wrong finish) of the parts.
3. All the parts must be labelled correctly for efficient management and to reduce human errors.

Defects	Proposed Action/Recommendation
Missing	Final QC at the end prior to packaging to ensure all parts are included
Wrong Style	Color code the barcodes or SKUs or some way to identify the different styles
Wrong Finish	Color code the barcodes or SKUs or some way to identify the different styles

In addition, some recommendations for the improvement on data collection for improved quality on the data models are key to the future analysis. The collection of categorical data should be more concise and grouped well. Some data was collected at parcel level and not at the style level which made it difficult to work with. Using the results of logical analysis and pairing them with experience to make informed decisions is the key to overall success in overall company's strategy.

6. REFERENCES

- Mid Term Report - Here
- https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e#:~:text=Feature%20Selection%20is%20the%20process,learn%20based%20on%20irrelevant%20features
- https://machinelearningmastery.com/what-is-imbalanced-classification/
- https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
- https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/
- https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html
- https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/
- https://scikit-learn.org/stable/modules/feature_selection.html
- Features selected for Methodology 2 - Here