Risk Classification & Machine learning  
safeWork risk analysis

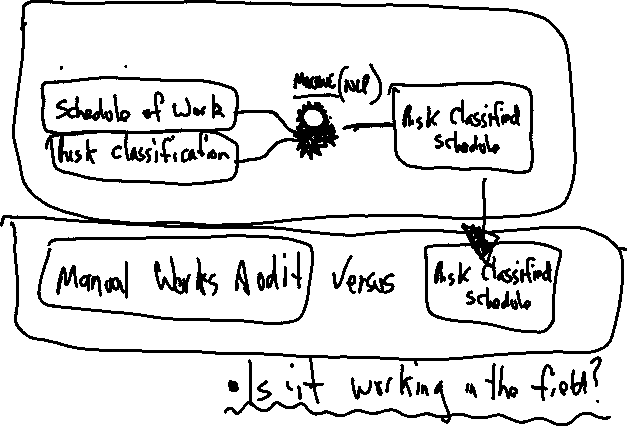
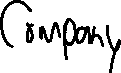
August 2021

**Team Members:** Josh Cullen

## Project Background and Description

Shape, circle

Description automatically generatedTo utilize a recent real time work audit to identify the specific job risk for a scope of work based on the description of work and discipline executing the scope of work.



## Risk Classification & Authority to Work Health Status

On any given day, there are over 400 contractor employees working on my job site, performing varying tasks such as welding, high voltage electrical work, civil, facility cleaning, High Pressure blasting to name a few. Each of these tasks require an Authority To Work authorisation prior to starting work and are risk classified appropriately based on the exposure to the business and the workers. For every job, a company representative is required to visit the job site with the contractor representative to ensure that there are no new hazards and the area is safe to start work using the existing approved controls.

The company representatives have identified that they are time poor with managing their own work crews let alone contractors and subsequently do not complete all tasks required of them with regards to both crew management (coaching, job checks, people management etc.) and Authority To Work requirements (field verifications, job front sign off).

In light of this concern, I employed a worker to come onsite and complete a Health Status audit on the Authority To Work process, where a contact was made with 386 work fronts to collect specific data based on the job and the Authority To Work form.

## Machine Learning (Natural Language Processing (NLP))

From the information gathered, we were able to develop a user defined risk classification tool which predicted whether the work to be performed was going to be either High Risk, Medium Risk or Low Risk.

Scikit-learn’s Naïve Bayes Classifier was used to predict the risk classifications.

## Library Installation

Text, letter

Description automatically generated

## Importing the data

We first start by reading in our data source and displaying the data

Table

Description automatically generated

## Cleaning the data

dropna() is used to drop the rows where at least one element is missing on the “Risk” & “Authority To Proceed Description” and then applied is a.str.lower() function to lowercase all characters in the new data frame.

Graphical user interface, text, application

Description automatically generated

## Tokenization

The Tokenization process is the used to extract the words in a sentence by spaces and punctuations. We use the nlkt [word\_tokenize](https://www.nltk.org/api/nltk.tokenize.html).

Also applied here is the str.lower() function to lowercase all characters in the new data frame.

Graphical user interface, text, application, email

Description automatically generated

## Tokenization Cleaning

In preparation to remove our STOP words, all non-alphanumeric characters were required to be removed. A function was created and the applied to the tokenized data frame.

Graphical user interface, text

Description automatically generated

## Remove STOP words

To increase the accuracy of the predict model, it is required to remove all STOP words that are not important in terms of the context, such as “the”, ”of”, “at”.

Graphical user interface, text, application, email

Description automatically generated

## Lemmatization

From the STOP words database, we now apply the Lemmatization function which is the process of making the same words in their stem. For example, weld, welding, welds are different in terms of Python. We lemmatiza the word to get weld.Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## Transform

A new column is created to encode our categories with our unique “Risk” values. The labelEncoder function is used here.

Graphical user interface, text

Description automatically generated

## Build the Model

Select our features (Authority To Proceed Description) and the target (risk\_encoded).



Split our data into train and test sets and use the stratify parameter of train\_test\_split since our data is unbalanced (more observations in one specific class than the others).



We can now get the tf-idf by using scikit-learn's TfidfVectorizer. This determines the most important words in the document.

Graphical user interface

Description automatically generated with medium confidence

The final build step to our machine learning model is to initialize the Multinomial Naive Bayes classifier, fit the model and check the accuracy.

Graphical user interface, application

Description automatically generated with medium confidence

## Model Evaluation

After generating the model, we use naïve bayes predict to check the accuracy using actual and predicted values.

Graphical user interface, application

Description automatically generated

## Finishing up with a dump

Store the model and vectorizer files for use when deploying form Heroku.

Graphical user interface, text, application, email

Description automatically generated