Development and Evaluation of Models for Coding Injury Cause Description

Natural Language Processing Fall 2022

Vitaliy Shkolnik (Shkolnik.v@northeastern.edu)
Lillith Chute (chute.l@northeastern.edu)
Adithya Abhishek Chenthilkannan (chenthilkannan.a@northeastern.edu)
GitHub Repo: https://github.com/vitaliy-shkolnik/nlp_insurance_project

***Abstract* - The worker's compensation industry contains a lot of challenges. One such challenge is finding an effective way to classify various forms of unstructured data. One such form deals with injury description classification. This project is designed to see if a model can be produced to effectively categorize injury cause descriptions using real world worker's compensation injury description claims. Frequently these descriptions are brief, and categories are non-standardized. As such, analyses were done around the length of text and its nature and examination of the categories. Based on this work, it was determined that there is a lot of overlap in categories. Further, that the descriptions were extremely short, frequently less than the size of a tweet. Given this, we chose to explore three different models. Naïve Bayes was chosen as a baseline. Second, that baseline was compared against Support Vector Machine and Random Forest/XGBoost. It was determined that XGBoost outperformed the others. However, there was not a statistically significant difference in the performance. Based on the findings, a major factor in the model's ability to predict is that the data needs to be significantly improved. Ideally, the data ingestion process could be changed such that overlapping categories would be combined. Several categories that have little to no data could be removed. Furthermore, descriptions would be constructed so that based on categories, key words would be included.**

Index Terms – Auto-coding data, Machine learning, Natural language Processing, Worker's compensation.

INTRODUCTION

The worker's compensation industry deals with a tremendous amount of unstructured data from images to video to various forms of text files. Most small to medium (likely many processes within large firms) deal with the processing of this data with a high degree of human touch. This significantly slows aspects of the workflow creating high wait times for the processing of claims to underwriting policies.

One such area of interest for the industry is in auto-coding of claims. This is the process of taking reported injuries and either fully medical coding them with ICD9 or ICD10 codes or taking smaller parts of that process and auto-coding categories based on free form text.

The problem we would like to address is attempting to accurately predict the injury cause based on short free form text descriptions. For instance, if the injury text was, "BEAM SLIPPED AND STRUCK EE'S RIGHT WRIST." The injury description would be correctly categorized as "Struck by falling object."

Having the ability to properly auto-code this via a natural language processing model would save considerable time in entering this data by hand. Further, it would eliminate potential human error up front in the workflow process. Given a highly accurate model, it would be used to generate a report on potential errors that have occurred in previously entered but active claims. Finally, at the backend of the workflow process, the Loss Control department that assesses risk would have much better data to work with.

Additionally, proving that this technique would work would provide evidentiary feedback that producing such models would be viable for other category classification tasks. Within the realm of claims and loss control, other categorizations that feed into the IDC9 or IDC10 are body part coding and nature of injury coding, to name a few.

BACKGROUND

Worker's compensation injury narratives, as noted, largely is comprised of unstructured text data. The need to be able to categorize such substantial amounts of textual data without having to manually classify is paramount to improving efficiency and accuracy.

Researchers in the past (Lehto, Marucci-Wellman, & Corns, 2009; Marucci-Wellman, Lehto, & Corns, 2011; Wellman, Lehto, & Sorock, 2004) have demonstrated that computer learning algorithms using Bayesian methods could auto-code injury descriptions into different causation groups, without any manual intervention, efficiently, and accurately. The authors demonstrated that the algorithms could code thousands of claims in a matter of minutes or hours with a high degree of accuracy by "learning" from claims previously coded by experts, subsequently to as a training set. Furthermore, these algorithms provided a score for each claim that reflected the algorithm's confidence in the prediction and, therefore, claims with low confidence scores could be flagged for manual review though this work did not attempt to calculate a confidence score.

Based on this research, the goal of this project was to develop and evaluate several auto-coding methods that could be used to aid the manual coding of the cause of injury for incoming claims. Other issues were investigated relating to the future implementation of the auto-coding method. Doing data analysis to determine the amount of separation of the classification categories. Varying the number of categories relative to the amount of data represented within each category type. The evaluation of varying hyperparameters. Finally, the performance of each model type selected for development. For the purposes of this project, the models being evaluated are Naïve Bayes, which will represent the baseline. Additionally, we will examine Support Vector Machine, Random Forest, and XGBoost.

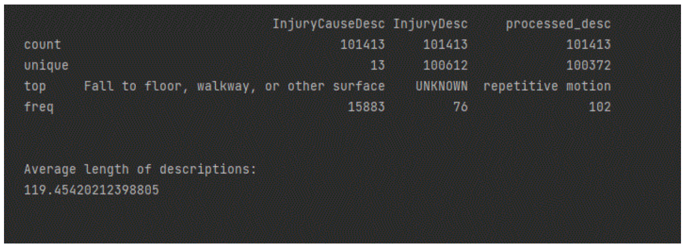
APPROACH

The approach we took to this project was to work through four phases. The first phase was to do some data analysis. This phase consisted of basic data examination, collecting basic metrics, and doing a T-SNE analysis of the categories for classification. The second phase was preprocessing. Based on the data analysis, we needed to implement appropriate forms of preprocessing and experiment with several types. Once we had the data cleaned up, we could move to model development. As noted above, we elected to try four model types. Naïve Bayes is a good baseline candidate for a natural language model. Once the baseline model is established, we work on the development of Support Vector Machine, Random Forest, and XGBoost models. After the completion of model development, we can evaluate each and do a comparison to determine which approach will work best.

DATA ANALYSIS AND VISUALIZATION

The original full data set that we were given to work with consisted of 235 separate categories with the number of injury cause descriptions ranging from 1 per category up to 15,884 for the top category. This initial examination, given the time constraints, made attempting to develop models across such drastic data imbalance impossible. Therefore, the first decision was to limit the classification to just the top 13 categories by row count. Once this was completed, there was still a significant data imbalance between the top category (15,884) and the bottom category (4,509). Also, eliminating all those other categories left us with about 101,428 rows of data to work with. This would eliminate working with deep learning models as they require substantial amounts of data to work with. Regardless, this was the data set we would work with.

Once this was established, we worked on getting some basic metrics. One of the items we were curious about was the average length of the descriptions we were working with. So, examining the raw data frame, we did some basic calculations and produced an average length of about 119 characters. This is less than half the maximum length of a tweet. This led to wondering if the descriptions would be of sufficient length to distinguish between closely related categories.



```
                                    InjuryCauseDesc InjuryDesc     processed_desc
count                                        101413     101413             101413
unique                                           13     100612             100372
top     Fall to floor, walkway, or other surface    UNKNOWN  repetitive motion
freq                                          15883         76                102


Average length of descriptions:
119.45420212398805
```

FIGURE I
BASE STATISTICS OF ALL 13 CATEGORIES

Based on the description length and the fact that we had a limited data set resulted in a closer examination of the categories. The reason for this becomes clear when reading them. For instance, the following categories of 'Struck against stationary object,' 'Struck by falling object,' and 'Struck by slipping handheld object' are all remarkably similar and would require the description to be truly clear for a model to properly classify it. This led to analysis of the categories. For this task we used T-SNE to check category separation.

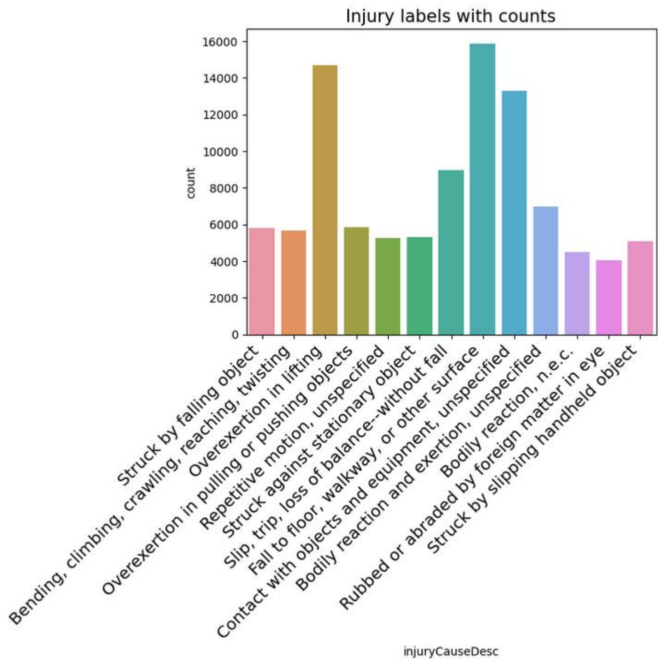


FIGURE II
DATA CATEGORY COUNTS

The data was also visualized in two dimensions to help gain a better understanding of how the categories related to each other. 200 data points from each label were randomly selected and plotted using the t-SNE module. It was revealed that two categories had poor clustering (Bodily reaction and exertion & unspecified & Bodily reaction, n.e.c.). This was further confirmed by the heatmap (Figure V) and correlated with our models' low predictions values. In contrast, the Rubbed or Abraded by foreign matter in eye had a very tight and isolated cluster. This result matched our model's high prediction rates. Finally, the t-SNE mapping confirmed that some categories have significant overlap, such as fall to floor and slip/trip. However, the map also revealed an overlap in categories we were not expecting (Struck by slipping handheld object + Contact with objects and equipment, unspecified).

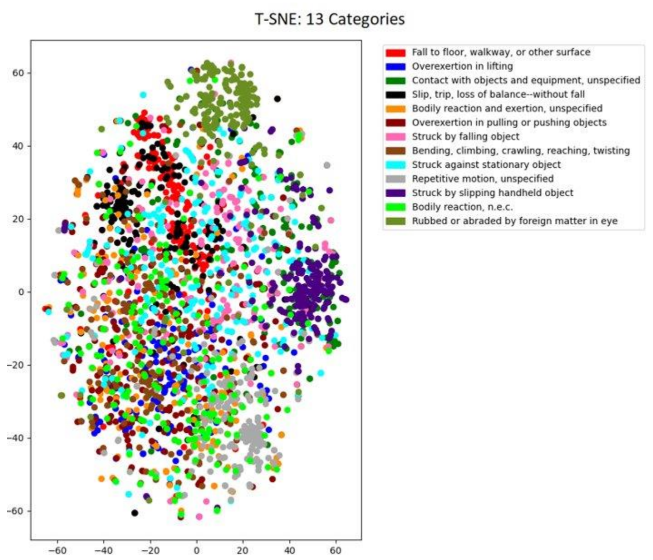


FIGURE III
T-SNE ANALYSIS OF ALL 13 CATEGORIES

PREPROCESSING

Examination of the description data showed many misspellings, abbreviations, random spacing, erratic punctuation, and often unclear language. Given the time constraints, we elected to do primarily basic preprocessing with a few additional features that we noticed while examining the data.

I. Basic

For the basic preprocessing steps, we wrote methods to lowercase text, remove numbers, remove punctuation, and white spaces. We added methods to replace contractions and remove URLs and non-ascii characters. We also experimented with converting numbers into text, but that did not seem to benefit the preprocessing, so we decided not to use it. Finally, we made sure we normalized the corpus, removed stop words, and lemmatized the verbs.

II. Over Sampling and Under Sampling

The injury coding data we found had mostly unbalanced classes. Sampling is an effective method to balance data. We performed experiments on oversampling and under sampling, the classes and found noticeable improvements in the result. For the first experiment we Under sampled the large classes and oversampled the minor classes to bring all classes to an approximate average count of records. The resulting models were partially only successful, as printing out the classification report pointed out that the oversampled classes performed poorly in the testing set compared to the training set. This was due to the overlap in the identical records created by oversampling the minority classes.

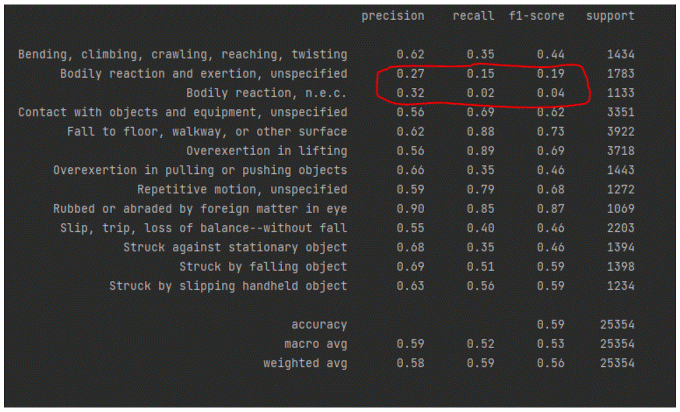
The next sampling approach tested was to retain the minor class numbers and under sample the major classes. This produced satisfactory results.

MODEL DEVELOPMENT

I. Baseline Naïve Bayes

The first model that we tackled was Naïve Bayes. This model was chosen for a couple of reasons. The first was that several of the auto-coding research papers that we have cited used this model as a comparatively well performing model given a less populous data set. Secondarily, Naïve Bayes is a well performing machine learning algorithm.

The first experiment was to see how well this model would perform with all thirteen categories without any hyperparameter tuning.



```
                                              precision    recall  f1-score   support

  Bending, climbing, crawling, reaching, twisting       0.62      0.35      0.44      1434
        Bodily reaction and exertion, unspecified       0.27      0.15      0.19      1783
                           Bodily reaction, n.e.c.       0.32      0.02      0.04      1133
  Contact with objects and equipment, unspecified       0.56      0.69      0.62      3351
         Fall to floor, walkway, or other surface       0.62      0.88      0.73      3922
                          Overexertion in lifting       0.56      0.89      0.69      3718
       Overexertion in pulling or pushing objects       0.66      0.35      0.46      1443
                   Repetitive motion, unspecified       0.59      0.79      0.68      1272
       Rubbed or abraded by foreign matter in eye       0.90      0.85      0.87      1069
        Slip, trip, loss of balance--without fall       0.55      0.40      0.46      2203
                 Struck against stationary object       0.68      0.35      0.46      1394
                         Struck by falling object       0.69      0.51      0.59      1398
                 Struck by slipping handheld object       0.63      0.56      0.59      1234

                                         accuracy                           0.59     25354
                                        macro avg       0.59      0.52      0.53     25354
                                     weighted avg       0.58      0.59      0.56     25354
```

FIGURE IV
NAÏVE BAYES MODEL STATISTICS

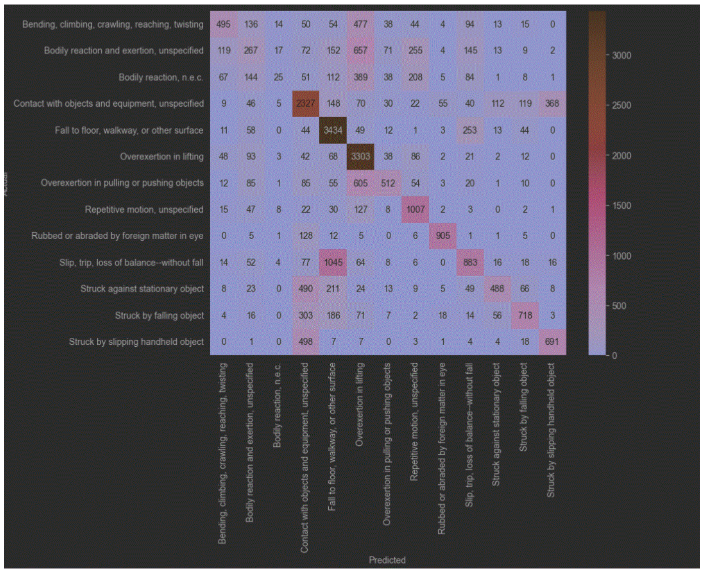
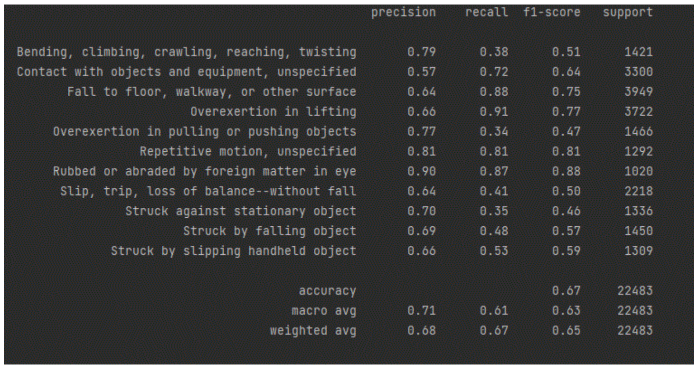


FIGURE V
NAÏVE BAYES MODEL HEAT MAP

In Figure IV, which represents the scoring across all 13 categories, there are two outlying categories that have minimal prediction viability. 'Bodily reaction and exertion, unspecified' and 'Bodily reaction, n.e.c.' both have extremely low F1 scores relative to the remaining 11 categories that we decided to throw those categories out. This is also reflected in the heat map as shown in Figure V.

After removing those two categories and re-running the model, the statistics improved considerably, from 56% to 65%, for the overall weighted F1 score among the remaining eleven categories as noted in Figure VI.



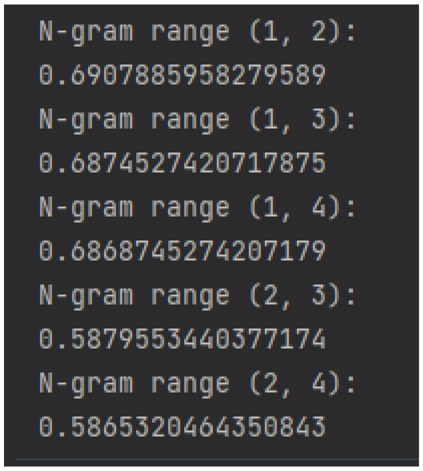
```
                                              precision    recall  f1-score   support

  Bending, climbing, crawling, reaching, twisting       0.79      0.38      0.51      1421
  Contact with objects and equipment, unspecified       0.57      0.72      0.64      3300
         Fall to floor, walkway, or other surface       0.64      0.88      0.75      3949
                          Overexertion in lifting       0.66      0.91      0.77      3722
       Overexertion in pulling or pushing objects       0.77      0.34      0.47      1466
                   Repetitive motion, unspecified       0.81      0.81      0.81      1292
       Rubbed or abraded by foreign matter in eye       0.90      0.87      0.88      1020
        Slip, trip, loss of balance--without fall       0.64      0.41      0.50      2218
                 Struck against stationary object       0.70      0.35      0.46      1336
                         Struck by falling object       0.69      0.48      0.57      1450
                 Struck by slipping handheld object       0.66      0.53      0.59      1309

                                         accuracy                           0.67     22483
                                        macro avg       0.71      0.61      0.63     22483
                                     weighted avg       0.68      0.67      0.65     22483
```

FIGURE VI
NAÏVE BAYES MODEL STATISTICS REMAINING CATEGORIES

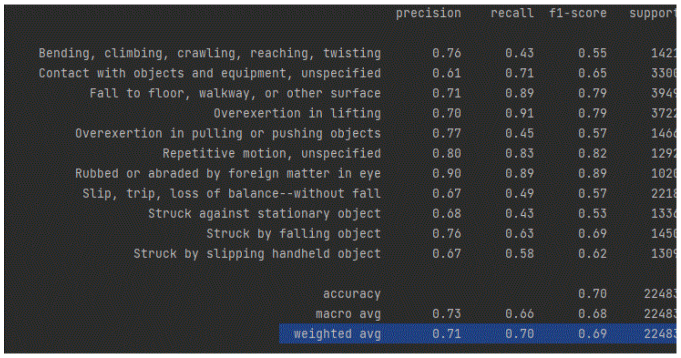
At this point, we moved on to tuning a couple of different hyperparameters. The first was working on n-gram ranges. We wanted to see what different ranges would do to the overall model prediction.



```
N-gram range (1, 2):
0.6907885958279589
N-gram range (1, 3):
0.6874527420717875
N-gram range (1, 4):
0.6868745274207179
N-gram range (2, 3):
0.5879553440377174
N-gram range (2, 4):
0.5865320464350843
```

FIGURE VII
N-GRAM RANGES AND ASSOCIATED ACCURACIES

As noted in Figure VII, the two best ranges were from (1, 2) to (1, 3). From there the next hyperparameter to explore was max features. This was explored in increments of 2,000 to 14,000 across both n-gram ranges. The result was that n-gram range (1, 2) combined with max features set to 4,000 produced the best overall result.

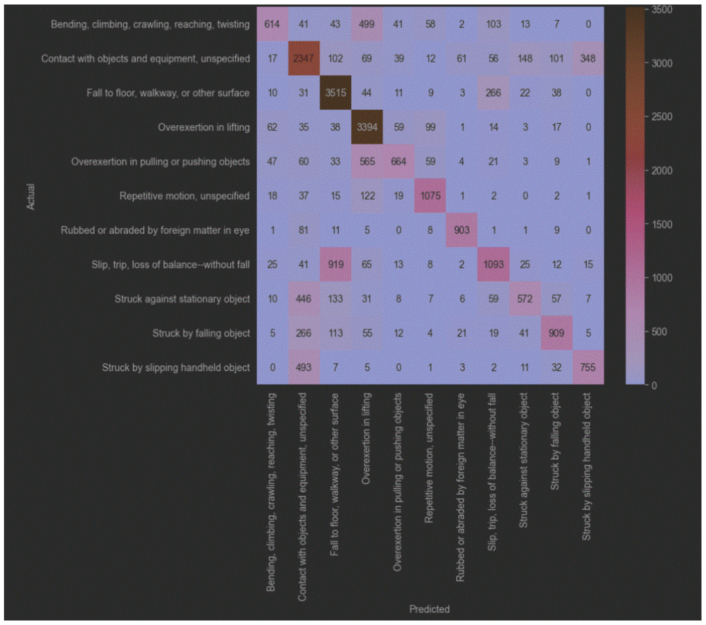


```
                                              precision    recall  f1-score   suppor

   Bending, climbing, crawling, reaching, twisting       0.76      0.43      0.55      1421
Contact with objects and equipment, unspecified       0.61      0.71      0.65      3306
       Fall to floor, walkway, or other surface       0.71      0.89      0.79      3949
                       Overexertion in lifting       0.70      0.91      0.79      3722
     Overexertion in pulling or pushing objects       0.77      0.45      0.57      1466
                Repetitive motion, unspecified       0.80      0.83      0.82      1292
     Rubbed or abraded by foreign matter in eye       0.90      0.89      0.89      1020
      Slip, trip, loss of balance--without fall       0.67      0.49      0.57      2218
             Struck against stationary object       0.68      0.43      0.53      1336
                      Struck by falling object       0.76      0.63      0.69      1450
           Struck by slipping handheld object       0.67      0.58      0.62      1309

                                      accuracy                          0.70     22483
                                     macro avg       0.73      0.66      0.68     22483
                                  weighted avg       0.71      0.70      0.69     22483
```

FIGURE VIII
MODEL STATISTICS WITH BEST HYPERPARAMETERS



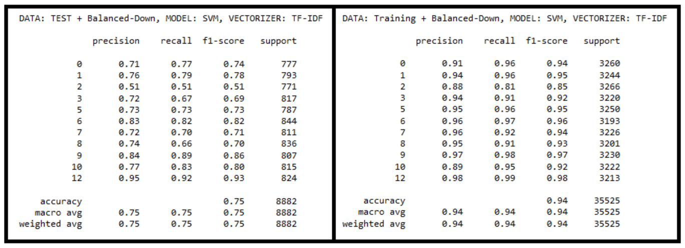
	Bending, climbing, crawling, reaching, twisting	Contact with objects and equipment, unspecified	Fall to floor, walkway, or other surface	Overexertion in lifting	Overexertion in pulling or pushing objects	Repetitive motion, unspecified	Rubbed or abraded by foreign matter in eye	Slip, trip, loss of balance--without fall	Struck against stationary object	Struck by falling object	Struck by slipping handheld object
Bending, climbing, crawling, reaching, twisting	614	41	43	499	41	58	2	103	13	7	0
Contact with objects and equipment, unspecified	17	2347	102	69	39	12	61	56	148	101	348
Fall to floor, walkway, or other surface	10	31	3515	44	11	9	3	266	22	38	0
Overexertion in lifting	62	35	38	3394	59	99	1	14	3	17	0
Overexertion in pulling or pushing objects	47	60	33	565	664	59	4	21	3	9	1
Repetitive motion, unspecified	18	37	15	122	19	1075	1	2	0	2	1
Rubbed or abraded by foreign matter in eye	1	81	11	5	0	8	903	1	1	9	0
Slip, trip, loss of balance--without fall	25	41	919	65	13	8	2	1093	25	12	15
Struck against stationary object	10	446	133	31	8	7	6	59	572	57	7
Struck by falling object	5	266	113	55	12	4	21	19	41	909	5
Struck by slipping handheld object	0	493	7	5	0	1	3	2	11	32	755

FIGURE IX
MODEL HEAT MAP

As noted in Figure VIII and IX, the F1 score improved from 65% to 69% with the tuned parameters and the accuracy went from 67% to 70.5%. This model then becomes the baseline model from which the others will be compared.

I. Support Vector Machine

Support Vector Machine (SVM) was tested using default settings and resulted in high prediction rates (Figure X). However, the results were suspicious, and the model was tested for overfitting using the training data. The results were significantly outside the 5% desired delta between test and training. Various hyperparameters were experimented with to reduce overfitting and it was determined that changing the kernel to linear and reducing the C from 1 to 0.2 (tradeoff between smooth decision boundary and classifying the training points correctly) created a model that offered high prediction values (76%), without overfitting the model (82%), refer to figure XI. This model warrants further testing to ensure true data predictions vs. learning how to predict one specific data set.



```
DATA: TEST + Balanced-Down, MODEL: SVM, VECTORIZER: TF-IDF

              precision    recall  f1-score   support

           0       0.71      0.77      0.74       777
           1       0.76      0.79      0.78       793
           2       0.51      0.51      0.51       771
           3       0.72      0.67      0.69       817
           5       0.73      0.73      0.73       787
           6       0.83      0.82      0.82       844
           7       0.72      0.70      0.71       811
           8       0.74      0.66      0.70       836
           9       0.84      0.89      0.86       807
          10       0.77      0.83      0.80       815
          12       0.95      0.92      0.93       824

    accuracy                           0.75      8882
   macro avg       0.75      0.75      0.75      8882
weighted avg       0.75      0.75      0.75      8882
```

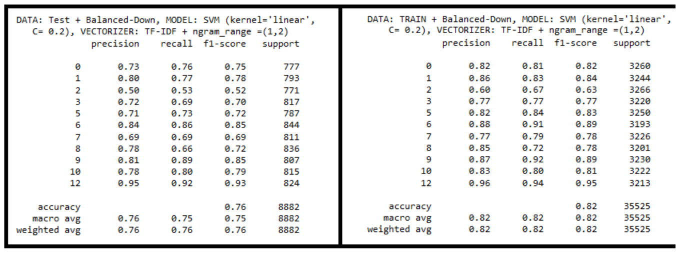
```
DATA: Training + Balanced-Down, MODEL: SVM, VECTORIZER: TF-IDF

              precision    recall  f1-score   support

           0       0.91      0.96      0.94      3260
           1       0.94      0.96      0.95      3244
           2       0.88      0.81      0.85      3266
           3       0.94      0.91      0.92      3220
           5       0.95      0.96      0.95      3250
           6       0.96      0.97      0.96      3193
           7       0.96      0.92      0.94      3226
           8       0.95      0.91      0.93      3201
           9       0.97      0.98      0.97      3230
          10       0.89      0.95      0.92      3222
          12       0.98      0.99      0.98      3213

    accuracy                           0.94     35525
   macro avg       0.94      0.94      0.94     35525
weighted avg       0.94      0.94      0.94     35525
```

FIGURE X
TRAIN / TEST COMPARISON DEFAULT HYPER-PARAMS



```
DATA: Test + Balanced-Down, MODEL: SVM (kernel='linear',
   C= 0.2), VECTORIZER: TF-IDF + ngram_range =(1,2)
              precision    recall  f1-score   support

           0       0.73      0.76      0.75       777
           1       0.80      0.77      0.78       793
           2       0.50      0.53      0.52       771
           3       0.72      0.69      0.70       817
           5       0.71      0.73      0.72       787
           6       0.84      0.86      0.85       844
           7       0.69      0.69      0.69       811
           8       0.78      0.66      0.72       836
           9       0.81      0.89      0.85       807
          10       0.78      0.80      0.79       815
          12       0.95      0.92      0.93       824

    accuracy                           0.76      8882
   macro avg       0.76      0.75      0.75      8882
weighted avg       0.76      0.76      0.76      8882
```

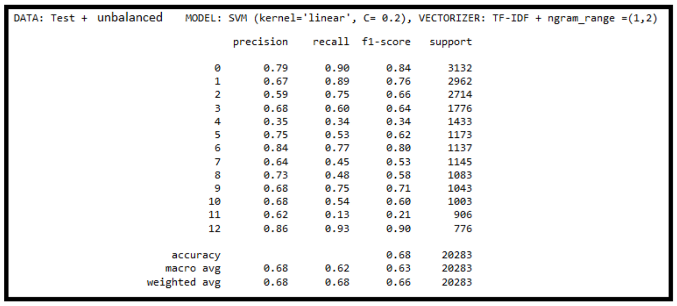
```
DATA: TRAIN + Balanced-Down, MODEL: SVM (kernel='linear',
   C= 0.2), VECTORIZER: TF-IDF + ngram_range =(1,2)
              precision    recall  f1-score   support

           0       0.82      0.81      0.82      3260
           1       0.86      0.83      0.84      3244
           2       0.60      0.67      0.63      3266
           3       0.77      0.77      0.77      3220
           5       0.82      0.84      0.83      3250
           6       0.88      0.91      0.89      3193
           7       0.77      0.79      0.78      3226
           8       0.85      0.72      0.78      3201
           9       0.87      0.92      0.89      3230
          10       0.83      0.80      0.81      3222
          12       0.96      0.94      0.95      3213

    accuracy                           0.82     35525
   macro avg       0.82      0.82      0.82     35525
weighted avg       0.82      0.82      0.82     35525
```

FIGURE XI

TRAIN / TEST COMPARISON FINE-TUNED HYPER-PARAMS



```
DATA: Test +  unbalanced   MODEL: SVM (kernel='linear', C= 0.2), VECTORIZER: TF-IDF + ngram_range =(1,2)

              precision    recall  f1-score   support

           0       0.79      0.90      0.84      3132
           1       0.67      0.89      0.76      2962
           2       0.59      0.75      0.66      2714
           3       0.68      0.60      0.64      1776
           4       0.35      0.34      0.34      1433
           5       0.75      0.53      0.62      1173
           6       0.84      0.77      0.80      1137
           7       0.64      0.45      0.53      1145
           8       0.73      0.48      0.58      1083
           9       0.68      0.75      0.71      1043
          10       0.68      0.54      0.60      1003
          11       0.62      0.13      0.21       906
          12       0.86      0.93      0.90       776

    accuracy                           0.68     20283
   macro avg       0.68      0.62      0.63     20283
weighted avg       0.68      0.68      0.66     20283
```

FIGURE XII

MODEL CREATED WITH ENTIRE DATA SET

I. Random Forest

The first process in developing the Random Forest model was to establish a baseline and understand how the data works with the algorithm at hand. The random forest model was initialized and run-on default parameters with 13 injury cause classes. The results unsurprisingly were not ideal, the test set performed poorly, and the training set overfit.

To solve these issues 3 main hyper-parameters were considered, Criterion, Max-depth, no-of estimators. By Iterating through multiple parameters, we found that the best criterion as 'Gini', and the max-depth need to be restricted down to 20, the no. of estimators is 150.

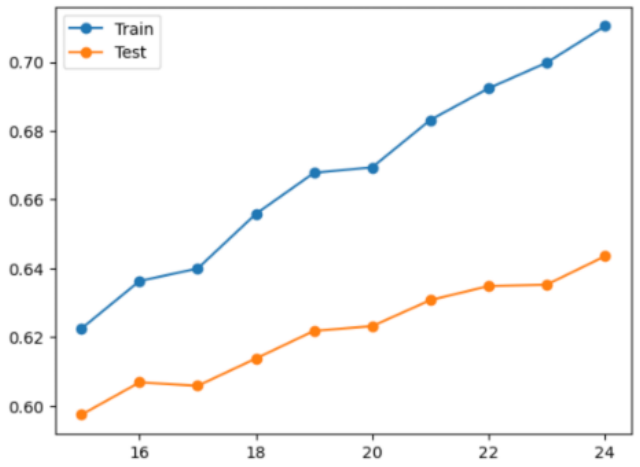


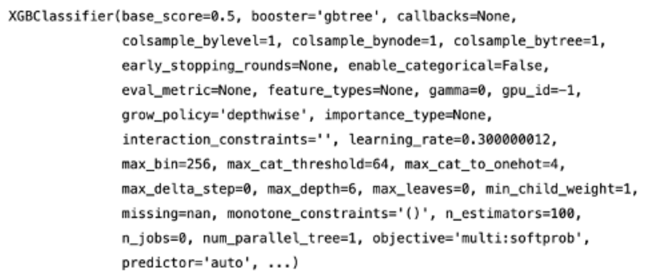
FIGURE XIII

TRAIN / TEST COMPARISON

The figure above shows the experiment that plots out accuracy score for max depth from 15 to 25. The final best parameters yielded a testing accuracy of 65% and a training accuracy of 70%.

I. XG Boost

The XG Boost model is an extremely popular and efficient gradient boosting model. The model attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. XG Boost model, because of its nature, is far better than the Random Forest model and achieves the best results very easily.

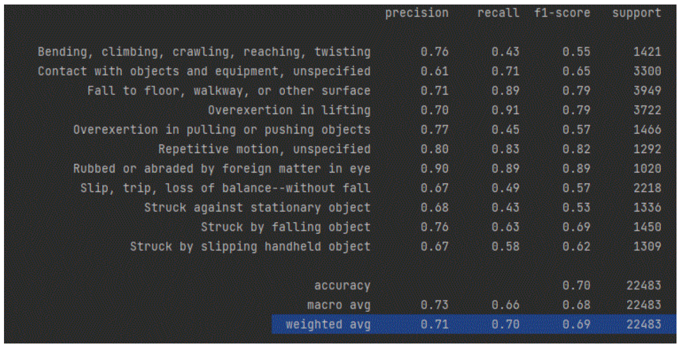


```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', ...)
```

The XG Boost model required much less effort to perform the best. The best parameters were remarkably close to the default.

RESULTS

Naïve bayes Results:



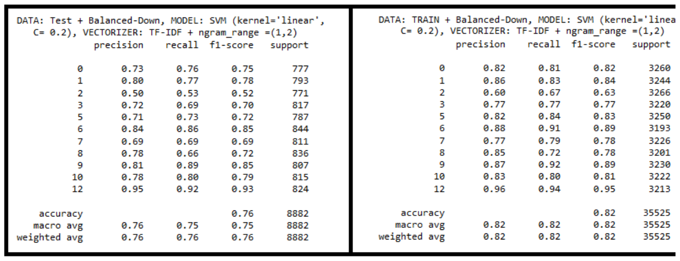
```
                                                precision    recall  f1-score   support

    Bending, climbing, crawling, reaching, twisting       0.76      0.43      0.55      1421
    Contact with objects and equipment, unspecified       0.61      0.71      0.65      3300
           Fall to floor, walkway, or other surface       0.71      0.89      0.79      3949
                             Overexertion in lifting       0.70      0.91      0.79      3722
       Overexertion in pulling or pushing objects       0.77      0.45      0.57      1466
                    Repetitive motion, unspecified       0.80      0.83      0.82      1292
       Rubbed or abraded by foreign matter in eye       0.90      0.89      0.89      1020
        Slip, trip, loss of balance--without fall       0.67      0.49      0.57      2218
                Struck against stationary object       0.68      0.43      0.53      1336
                         Struck by falling object       0.76      0.63      0.69      1450
                Struck by slipping handheld object       0.67      0.58      0.62      1309

                                          accuracy                          0.70     22483
                                         macro avg       0.73      0.66      0.68     22483
                                      weighted avg       0.71      0.70      0.69     22483
```

FIGURE XIV
FINAL MODEL RESULT NAÏVE BAYES

SVM Results:



```
DATA: Test + Balanced-Down, MODEL: SVM (kernel='linear',
   C= 0.2), VECTORIZER: TF-IDF + ngram_range =(1,2)
              precision    recall  f1-score   support

           0       0.73      0.76      0.75       777
           1       0.80      0.77      0.78       793
           2       0.50      0.53      0.52       771
           3       0.72      0.69      0.70       817
           5       0.71      0.73      0.72       787
           6       0.84      0.86      0.85       844
           7       0.69      0.69      0.69       811
           8       0.78      0.66      0.72       836
           9       0.81      0.89      0.85       807
          10       0.78      0.80      0.79       815
          12       0.95      0.92      0.93       824

    accuracy                           0.76      8882
   macro avg       0.76      0.75      0.75      8882
weighted avg       0.76      0.76      0.76      8882
```

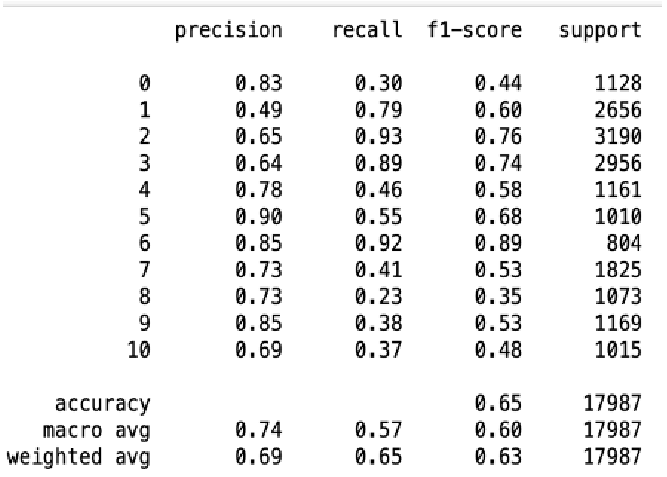
```
DATA: TRAIN + Balanced-Down, MODEL: SVM (kernel='linea
   C= 0.2), VECTORIZER: TF-IDF + ngram_range =(1,2)
              precision    recall  f1-score   support

           0       0.82      0.81      0.82      3260
           1       0.86      0.83      0.84      3244
           2       0.60      0.67      0.63      3266
           3       0.77      0.77      0.77      3220
           5       0.82      0.84      0.83      3250
           6       0.88      0.91      0.89      3193
           7       0.77      0.79      0.78      3226
           8       0.85      0.72      0.78      3201
           9       0.87      0.92      0.89      3230
          10       0.83      0.80      0.81      3222
          12       0.96      0.94      0.95      3213

    accuracy                           0.82     35525
   macro avg       0.82      0.82      0.82     35525
weighted avg       0.82      0.82      0.82     35525
```

FIGURE XV
TRAIN / TEST COMPARISON FINE-TUNED HYPER-PARAMS

Random Forest Results:



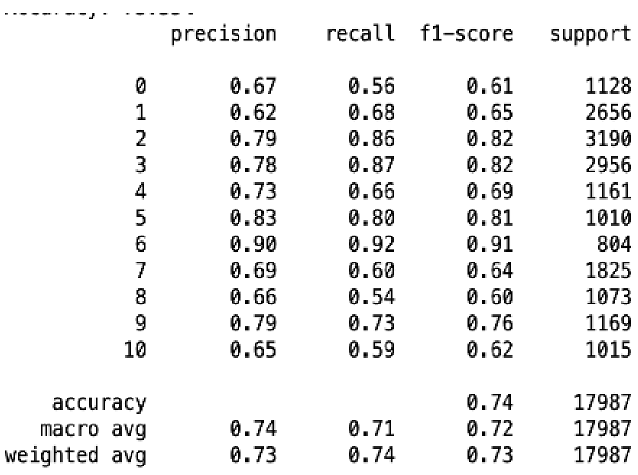
```
              precision    recall  f1-score   support

           0       0.83      0.30      0.44      1128
           1       0.49      0.79      0.60      2656
           2       0.65      0.93      0.76      3190
           3       0.64      0.89      0.74      2956
           4       0.78      0.46      0.58      1161
           5       0.90      0.55      0.68      1010
           6       0.85      0.92      0.89       804
           7       0.73      0.41      0.53      1825
           8       0.73      0.23      0.35      1073
           9       0.85      0.38      0.53      1169
          10       0.69      0.37      0.48      1015

    accuracy                           0.65     17987
   macro avg       0.74      0.57      0.60     17987
weighted avg       0.69      0.65      0.63     17987
```

FIGURE XVI
RANDOM FOREST RESULTS

XG Boost Results:



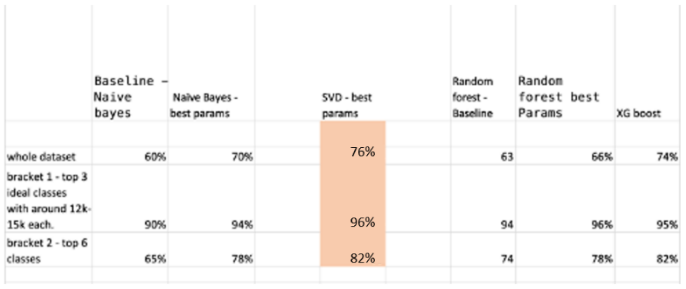
```
              precision    recall  f1-score   support

           0       0.67      0.56      0.61      1128
           1       0.62      0.68      0.65      2656
           2       0.79      0.86      0.82      3190
           3       0.78      0.87      0.82      2956
           4       0.73      0.66      0.69      1161
           5       0.83      0.80      0.81      1010
           6       0.90      0.92      0.91       804
           7       0.69      0.60      0.64      1825
           8       0.66      0.54      0.60      1073
           9       0.79      0.73      0.76      1169
          10       0.65      0.59      0.62      1015

    accuracy                           0.74     17987
   macro avg       0.74      0.71      0.72     17987
weighted avg       0.73      0.74      0.73     17987
```

FIGURE XVII
XG BOOST RESULTS

Ablation Table:



	Baseline – Naive bayes	Naive Bayes - best params	SVD - best params	Random forest - Baseline	Random forest best Params	XG boost
whole dataset	60%	70%	76%	63	66%	74%
bracket 1 - top 3 ideal classes with around 12k-15k each.	90%	94%	96%	94	96%	95%
bracket 2 - top 6 classes	65%	78%	82%	74	78%	82%

FIGURE XIII
FINAL MODEL COMPARISON

CONCLUSION

As noted in Figure XIII, after comparing the overall performance of our four tuned models, XG Boost performed best across the initial 11 categories as well as in the top 6 category bracket. It fell just 1% shy of equaling Random Forest in the top 3 category bracket. Another thing worth noting is that, relatively speaking, there is not a great deal of accuracy separation across the different models, except for Random Forest in the case of the whole data set or in the top six category bracket. It is also important to mention that in the case of SVD, it was overfitting more than XG Boost, which gives the edge to XG Boost in the event of similar accuracies.

- Preliminary investigations of SVM show very promising performance, but the model is prone to overfitting.

Decreasing the C hyperparameter (tradeoff between smooth decision boundary and classifying the training points correctly) appeared to resolve the overfitting problem, but further investigation and evidence is needed before confirmation (figure XI). In addition, this model required the data to be balanced and providing the model significantly unbalanced data resulted in poor performance (figure XII).

There are several additional points worth noting because of this work. The first point is a result of the data analysis work. As relayed in the data analysis section, there are 235 separate categories of injury cause. We should note that just in the top 13 categories we analyzed, there is a significant overlap. This coupled with the fact that the descriptions are extremely short and frequently vague indicates that the data ingestion process has a lot of room for improvement. Ideas for improvement are combining related categories as indicated via clustering algorithms. Using key words as part of the descriptions as they are entered to help strongly correlate a description with a classification. Further, constructing a keyword dictionary to use in conjunction with the description.

Another avenue to explore would be the idea of chaining more specifically focused models. The idea here would be to train smaller models focused on specific groupings of categories. For instance, for all the ‘struck by an object’ variants, we create a specific model to inject that set of data. This way we create a suite of models to deal with categorical groupings to pass injury cause descriptions through. The hope is that it would result in overall higher accuracy.

If we had more time to invest in this project, we would like to further explore the description data itself and try out further preprocessing techniques to see if there are more techniques to aid predictability. We might further do analysis to see if we can find higher correlated descriptions to see why that is and how we might use that information. Finally, we would like to attempt some of the hypotheses we indicated above.

References

[1] Bertke SJ, Meyers AR, Wurzelbacher SJ, Bell J, Lampl ML, Robins D. Development and evaluation of a Naïve Bayesian model for coding causation of workers' compensation claims. J Safety Res. 2012 Dec;43(5-6):327-32.

[2] Bertke SJ, Meyers AR, Wurzelbacher SJ, Measure A, Lampl MP, Robins D. Comparison of methods for auto-coding causation of injury narratives. Accid Anal Prev. 2016 Mar; 88:117-23.

[3] Stanfill MH, Williams M, Fenton SH, Jenders RA, Hersh WR. A systematic literature review of automated clinical coding and classification systems. J Am Med Inform Assoc. 2010 Nov-Dec;17(6):646-51.

[4] Wellman, Helen & Corns, Helen & Lehto, Mark. (2017). Classifying injury narratives of large administrative databases for surveillance—A practical approach combining machine learning ensembles and human review. Accident Analysis & Prevention. 98. 359-371.

[5] Wellman, Helen & Lehto, Mark & Corns, Helen. (2015). A practical tool for public health surveillance: Semi-automated coding of short injury narratives from large administrative databases using Naïve Bayes algorithms. Accident; analysis and prevention. 84. 165-176.