UNIVERSITÀ DI PISA

Data Mining and Machine Learning

Construction-Accident Report Classification

Andrea Vagnoli 635788 a.a. 2024 - 2025

Indice

1 Introduction

According to a report by the International Labour Organization (ILO), around 2.78 million workers die each year due to occupational accidents, while approximately 374 million experience nonfatal injuries that result in at least four days of work absence. These incidents not only lead to severe human suffering but also generate significant economic costs, amounting to nearly 3.94% of the global GDP annually. Due to complex and often inadequate safety management, the construction industry accounts for one out of every six fatal workplace accidents, marking it as one of the most dangerous sectors.

Following an accident, detailed reports are usually compiled, which include both structured data (e.g., date of the incident, company information, and demographics of the injured individuals) and unstructured narrative data (e.g., descriptions and summaries of the event). These reports often include findings from investigations, such as specific safety violations. However, their unstructured nature poses considerable challenges for analysis and knowledge extraction.

Identifying and leveraging insights from historical accident data is therefore essential for informing proactive safety strategies and effective risk prevention in the future.

In this context, the objective of this project is to develop a classification model capable of assigning construction accident reports to their correct category using traditional methods of text vectorization and more advanced methods such as word embeddings.

2 Related Works

The dataset used in the present study originates from a recent and relevant work by Qiao et al. (2022) [2], which reports on the classification of reports into seven distinct categories identified by the authors through the use of a standardized labeling system known as the Occupational Injury and Illness Classification System (OIICS). In their study, the authors achieved the best results—with an F1 score of 0.85 and a weighted F1 score of 0.91—using text vectorization methods combined with classical machine learning approaches. These values represent a significant performance improvement compared to previous studies in this context, making their method one of the most effective in identifying the causes of construction-related incidents. According to the authors, this improvement is primarily due to a revised and more accurate labeling of incidents, using more appropriate categories and causes.

However, despite these strong aggregate results, the authors highlight challenges in recognizing minority classes with lower F1 scores in the rarest categories. These findings indicate a clear performance imbalance between frequent and infrequent classes.

One major challenge reported in the study is that some reports belonging to minority classes may involve multiple contributing factors. Nevertheless, classification—according to the OIICS standard—is based solely on the primary cause, which can be difficult to determine, especially when the narratives contain overlapping or ambiguous descriptions of the incidents.

3 Dataset

This dataset contains 4,770 construction accident reports from the Occupational Safety and Health Administration (OSHA) and is available **here**. The dataset is taken from the work previously referenced [2], and it includes the following fields:

- **id**: A numerical code that identifies the current record;
- **title**: A short description of the accident report;
- **SUMMARY**: A narrative text describing the details of the accident;
- **TggedL1**: This is the primary label of the accident. There are seven distinct categories:
  - `TRANSPORTATION INCIDENTS`
  - `FIRES AND EXPLOSIONS`
  - `FALLS, SLIPS, TRIPS`
  - `EXPOSURE TO HARMFUL SUBSTANCES OR ENVIRONMENTS`
  - `CONTACT WITH OBJECTS AND EQUIPMENT`
  - `OVEREXERTION AND BODILY REACTION`
  - `VIOLENCE AND OTHER INJURIES BY PERSONS OR ANIMALS`

  These categories follow the *Occupational Injury and Illness Classification System (OIICS)* Level 1 classification to prevent ambiguity. The OIICS defines only seven high-level categories, and standard labeling rules were applied. Labels were assigned by domain experts to minimize the risk of misclassification.
- **TggedL2**: These are secondary labels providing additional details about the accidents. However, they will not be considered in this study.

For the purpose of this work, following also the same strategy as in the original paper, the `title` and `SUMMARY` columns were merged into a single textual field, while only the `TggedL1` column was retained as the classification target. All other columns were discarded.

As a result, the final dataset used for the experiments consists of 4,770 rows and two columns: a textual column called `Report` containing the combined `title + SUMMARY`, and the corresponding `TggedL1` label called `Label`.

4 Exploratory Analysis

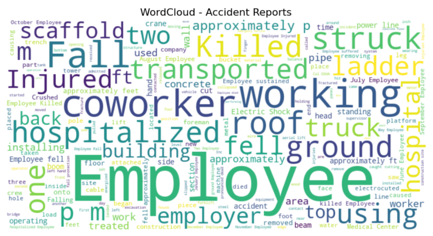


Figure 1: WordCloud of the Report column

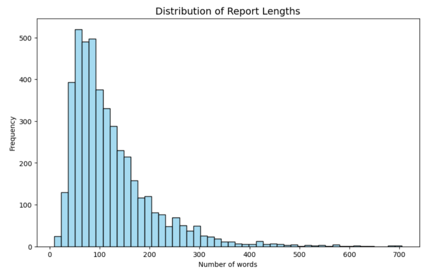


Figure 2: Word distributions of the Report column

In Figure 1, we can observe the word cloud generated from the textual reports, which highlights the presence of terms typically associated with the construction and injury context. In addition to Figure 2, which shows the distribution of report lengths, it is also important to consider Figure 3, which presents the class distribution within the dataset. As previously mentioned, we are dealing with a highly imbalanced classification problem, where the two minority classes have a ratio of approximately 0.03 and 0.008 compared to the majority class. This imbalance must be carefully taken into account in the continuation of the work.

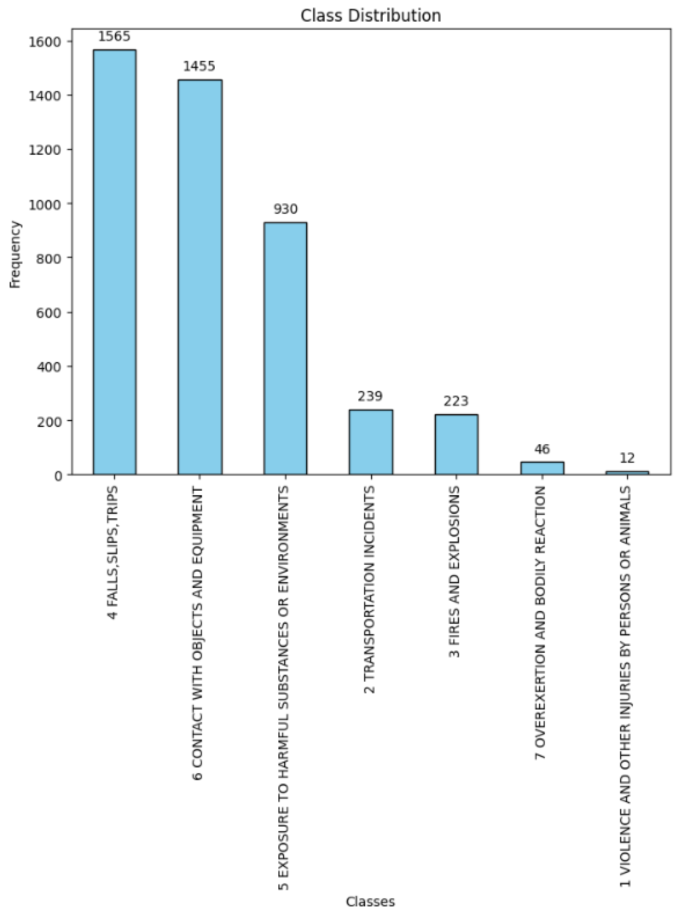


Figure 3: Class Distribution

At this point, we split our data into 70% for the training set and 30% for the test set.

5 Data Preprocessing

For this dataset, in order to use it with word embedding and text vectorization tecniques we will apply the following steps:

- **Tokenization**;
- Convert to **lowercase**;
- Removal of **punctuation** and **stopwords**;

- **Stemming**: Each token is processed using a Stemmer, which reduces words to their root form. A stemmer was chosen over a lemmatizer to prioritize simplicity and computational efficiency.

6 Text Representation

At this stage of the analysis, we need to convert the textual format of the data into a numerical format that can be interpreted by our classification models. The choice to use two different types of text representation models was mainly driven by a previously mentioned issue in the paper from which the data originates. Specifically, it was found that some classification errors were due to the fact that certain reports appeared to contain more than one single cause of the incident, leading to partial label overlap. According to the labeling model used (OICSS), the cause must be singular, assigning to each report the primary cause of the incident or, alternatively, the cause that occurred chronologically first. In an effort to address this, we decided to explore two techniques: word embedding and text vectorization.

6.1 Word Embedding

Word Embedding can be particularly useful in addressing the issue of partial labeling present in our dataset. Word2Vec is a powerful technique in Natural Language Processing (NLP) that learns distributed vector representations of words by capturing their semantic meaning based on surrounding context. By mapping words into a continuous vector space where semantically similar terms are placed closer together, it enables the model to better understand relationships and meaning. This, in turn, may help in identifying the most relevant cause when multiple causes are mentioned within the same report.

6.1.1 Word2Vec

We used the Word2Vec implementation from the `gensim` library to generate word embeddings. This tool is widely adopted in NLP tasks and provides an efficient and flexible interface for training custom embeddings on domain-specific data. We started with commonly used parameters suitable for our task:
Parameter:

- `vector_size=100:` Sets the dimension of word vectors. 100 is a balanced choice for capturing semantic meanings.
- `window=5:` Defines the context size around a target word. A window of 5 ensures enough context is considered without being too broad.
- `min_count=1:` Includes all words, even rare ones, which is useful in datasets with domain-specific or sparse vocabulary.
- `sg=1:` Uses the Skip-gram model, which is effective at capturing relationships between rare words and their contexts.

6.2 Text Vectorization

6.2.1 WCTF-IDF (Weighted Class TF-IDF)

In the analysis mentioned earlier, from which the data originates, the most performant classification model uses TF-IDF as the textual representation of the data. In that study it has been observed that the model has more difficulty to recognize the two minority classes due to the fact that, as previously stated, overlapping causes emerge in the incident reports, causing the classification model to identify the wrong cause because of the higher weight of words from the majority class. Furthermore, following the classic TF-IDF approach, it is necessary to use a very high number of features so that the vocabulary includes the very rare words from the minority classes (in the previous example, the most performant model reports 8423 features compared to the dataset's total of 4470 samples). For this reason, we will use a variant of the classic TF-IDF model.
This variant of TF-IDF is a recently proposed approach [3]. In a context of imbalanced classes, as is the case in this work, it has emerged that standard TF-IDF often underrepresents the minority class, since it selects features solely based on overall term frequency. This results in the majority class dominating the feature space.

Vanilla TF-IDF computes the importance of a term x in a document y using:

$$TF-IDF(x,y) = tf(x,y) \cdot \log\left(\frac{N}{df(x)}\right)$$

where $tf(x,y)$ is the term frequency of word x in document y, N is the total number of documents, and $df(x)$ is the number of documents containing x.

However, this approach favors frequent terms across the corpus, which often belong to the majority class. To counteract this, Weighted Class TF-IDF (WCTF-IDF) selects features proportionally from each class. If n_{ci} is the number of documents in class i and n_d is the total number of documents, and we want to select f features overall, the number of features f_i allocated to class i is:

$$f_i = f \cdot \frac{n_{ci}}{n_d}$$

TF-IDF is then computed separately on each class using f_i features, and the final vocabulary is the union of the class-specific vocabularies. This vocabulary is then used in a global TF-IDF transformation over the entire dataset.

This method ensures a fair representation of both majority and minority classes in the feature space, trying to improve model performance in imbalanced multiclass scenarios.

Further, in order to ensure a non-overlapping feature set between classes, features selected from the class with a larger number of samples are passed as stop-words to the stop_words hyperparameter when running TF-IDF over the class with a smaller number of samples. In this case of multiclass problems, this procedure is performed in the order of the class sizes, i.e., starting with the class that has the largest number of samples.

We have decided to put the initial value of the `max_features` parameter in the TF-IDF vectorizer was initially set to 1000 to limit dimensionality and reduce the risk of overfitting.

7 Oversampling

7.1 SMOTE

Once we have decided on the text representation of our input data, we must take into account the fact that the multiclass problem we are facing is highly imbalanced. To address this, the Synthetic Minority Over-sampling Technique (SMOTE) is employed as an oversampling method. SMOTE generates synthetic examples of the minority classes by interpolating between existing samples, effectively balancing the dataset without simply duplicating entries. This approach helps mitigate the bias towards majority classes during model training, allowing the classifier to better learn the decision boundaries of underrepresented categories.

8 Model Selection

During this phase, our objective is to analyze various classification models from the classical machine learning approach and select the most performant model using a predefined criterion. In the case of a heavily imbalanced multiclass classification problem, the most useful metric is the weighted F1-SCORE, which also takes into account the multiplicity of individual samples per class. The models we will consider are:

- SVM (linear kernel);
- Logistic regression;
- K-nearest neighbors;
- Naive Bayes;
- XGBoost;
- Bagging;
- Decision Tree;
- Random Forest;

To evaluate the performance of each model optimized through grid search, we will use 5-fold nested cross-validation on the training set (for the tuning parameters used for each model, please refer to the code). This approach allows us to compare the models, already tuned via hyperparameter search, using both the word-embedding technique and the weighted class TF-IDF method.

8.1 Pipeline

The workflow pipeline that is implemented each time on the training data is as follows: **pre-processing**, **text representation** (word2Vec or WCTF-IDF), oversampling with **SMOTE**, and **training** of the current classification model.

8.2 Results

Table 1: Results of 5-fold cross-validation: Accuracy and weighted F1-score

Model	**WCTF-IDF**		**Word2Vec**	
	Accuracy	**Weighted F1-score**	**Accuracy**	**Weighted F1-score**
Random Forest	0.883 ± 0.007	0.879 ± 0.007	0.817 ± 0.010	0.815 ± 0.012
Logistic Regression	$\mathbf{0.903 \pm 0.007}$	$\mathbf{0.903 \pm 0.007}$	0.808 ± 0.005	0.818 ± 0.003
Linear SVM	0.901 ± 0.008	0.901 ± 0.008	0.820 ± 0.009	0.826 ± 0.007
XGBoost	0.891 ± 0.006	0.889 ± 0.006	0.826 ± 0.008	0.825 ± 0.007
Bagging	0.824 ± 0.015	0.829 ± 0.012	0.786 ± 0.019	0.787 ± 0.017
Decision Tree	0.794 ± 0.014	0.798 ± 0.013	0.680 ± 0.016	0.688 ± 0.015
KNN	0.680 ± 0.007	0.705 ± 0.008	0.727 ± 0.013	0.744 ± 0.012
MultinomialNB	0.856 ± 0.017	0.856 ± 0.016	—	—

Based on the results obtained, the best-performing method in terms of weighted F1-score is SVM when using WCTF-IDF. Moreover, with the exception of KNN, all models show better performance with WCTF-IDF. This suggests that WCTF-IDF may provide a more effective representation of the problem compared to word embeddings.

8.3 Statistical Comparison

The pairwise statistical comparison of all models would be computationally too demanding. For this reason, we will conduct two tests:

- The first test compares the two best-performing models overall, namely Logistic Regression and Linear SVM, both using the WCTF-IDF representation. The goal is to determine whether the observed difference in performance is statistically significant or not, and ultimately decide which of the two models is more suitable for the final deployment;

- The second test focuses on comparing Linear SVM trained with WCTF-IDF versus Linear SVM trained with word embedding. Since this is the same algorithm evaluated with two different text representations, this test is designed to provide further evidence that WCTF-IDF offers a better representation of the problem than word embeddings in our specific setting.

To obtain the most reliable comparison, each test is repeated using 5-fold cross-validation run six times independently.

8.3.1 SVM vs LR

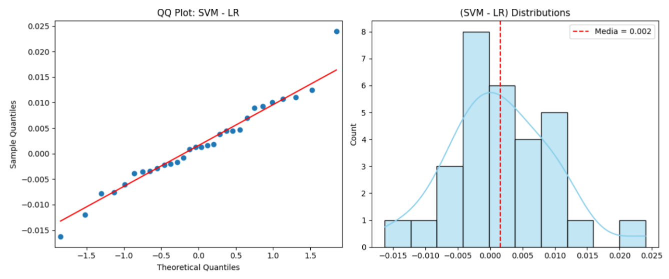


Figure 4: QQ Plot and Distribution

First, I verified whether the distribution was normal by using Q-Q plots and histograms, as shown in the Figure. As we can observe, the differences did not appear to be normally distributed, so I opted for a Wilcoxon signed-rank test instead. The Wilcoxon test returned a p-value of 0.36; therefore, the null hypothesis cannot be rejected. This implies that the two models cannot be explicitly recognized as one being more performant than the other.

8.3.2 SVM WCTF-IDF vs SVM word2vec

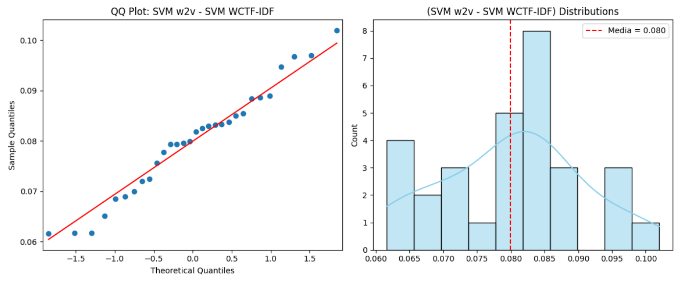


Figure 5: QQ Plot and Distribution

Also in this case, no Gaussianity emerges, so I proceed with the Wilcoxon signed-rank test. The Wilcoxon test returned a p-value of 0.0000000019, thus providing statistical evidence that, given the same model, the WCTF-IDF method appears to perform better for this task.

9 Performance Evaluation

9.1 Test Set

To assess the generalization ability of our two models on unseen data, we evaluate their classification performance on the test set.

Model	**Parameter**	**Range Tested**	**Best Value**
Logistic Regression	`C`	[0.1, 1, 10]	1
Logistic Regression	`solver`	['liblinear', 'saga']	'saga'
Logistic Regression	`max_iter`	[100, 300, 1000]	100
SVM (Linear)	`C`	[0.1, 1, 10]	1

Table 2: Hyperparameter tuning results for Logistic Regression and Linear SVM

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 1	1.00	0.50	0.67	4
Class 2	0.70	0.76	0.73	72
Class 3	0.93	0.94	0.93	67
Class 4	0.95	0.92	0.93	469
Class 5	0.93	0.95	0.94	279
Class 6	0.88	0.88	0.88	436
Class 7	0.83	0.71	0.77	14
Accuracy		0.90		1341
Macro Average	0.89	0.81	0.83	1341
Weighted Average	0.90	0.90	0.90	1341

Table 3: Classification report SVM on the test set.

As we can see from Tables 3 and 4, the two models are practically equivalent in terms of summary metrics: accuracy, and macro and micro averages of Precision, Recall, and F1 score are the same. These values are high for every metric. The difference in F1 score at the single-class level appears in the two minority classes, namely Class 1 and Class 7. However, as we can also see from the respective confusion matrices, it should be noted that this difference has a relative meaning since the support for these two classes is very low, so misclassifications—even of a single label—can strongly impact the Precision or Recall values, and consequently the F1 score.

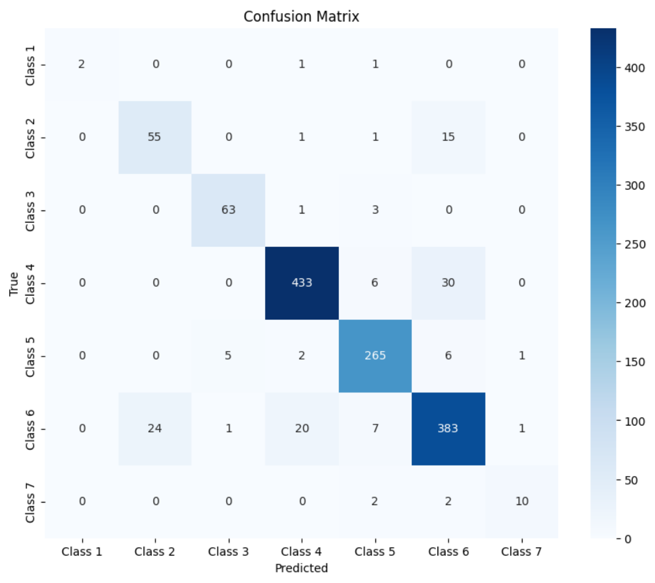


Figure 6: Confusion Matrix SVM

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Class 1	1.00	0.75	0.86	4
Class 2	0.72	0.81	0.76	72
Class 3	0.91	0.96	0.93	67
Class 4	0.94	0.92	0.93	469
Class 5	0.95	0.95	0.95	279
Class 6	0.89	0.88	0.89	436
Class 7	0.65	0.79	0.71	14
Accuracy		0.91		1341
Macro avg	0.87	0.86	0.86	1341
Weighted avg	0.91	0.91	0.91	1341

Table 4: Classification report LR on the test set.

9.2 Model comparison with other studies

At this point, we can compare our results with those obtained in the paper from which the dataset originates. The best-performing model in the paper is an SVM using TF-IDF as the text representation method, with a feature size of 8,423. This model reports an accuracy of 0.91 and a weighted F1-score of 0.91, which are in line with our results. Notably, our model uses fewer than 1,000 features, which could indicate an improvement in feature selection—potentially

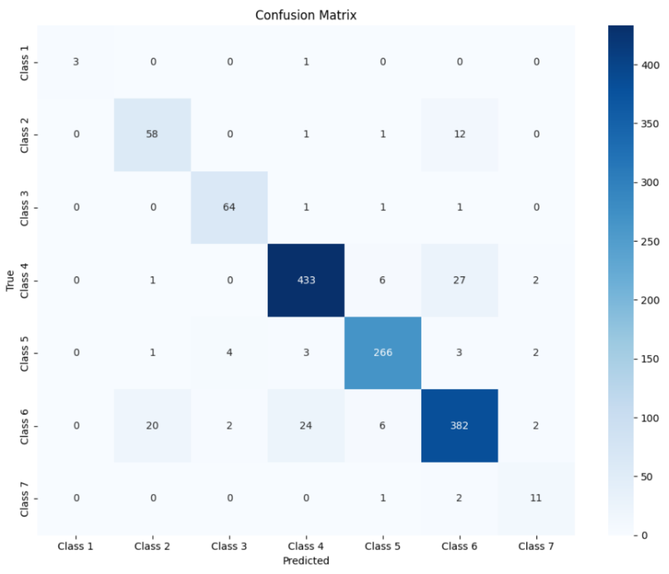


Figure 7: Confusion Matrix Logistic Regression

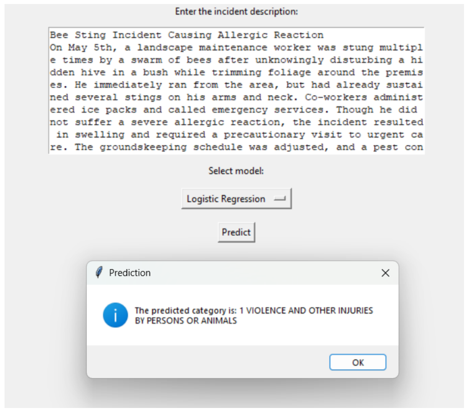
avoiding features that merely introduce noise—since the results remain comparable despite the reduced dimensionality.

Moreover, the paper reports F1-scores for classes 1 and 7 as 0.40 and 0.62, respectively, while our best-performing models achieved 0.86 and 0.77 for the same classes. These improvements might be attributed to the use of WCTF-IDF as a text representation instead of standard TF-IDF. However, due to the low number of test samples in these two classes—caused by strong class imbalance—these results should be interpreted with caution, as small variations can significantly affect the scores.

We can therefore conclude that, in a preliminary analysis, our trained model appears to be comparable to the one presented in the paper, while also showing potential improvements in both feature efficiency and performance on imbalanced classes.

10 Interface

To test the models with the best results, I created a simple interface in Python using the Tkinter library. The interface allows the user to write a report and choose which model to use between SVM and LR. By clicking the Predict button, the interface calls the predict method on the selected saved model. The graphical interface is shown in Figure 9.


Enter the incident description:

```
Bee Sting Incident Causing Allergic Reaction
On May 5th, a landscape maintenance worker was stung multipl
e times by a swarm of bees after unknowingly disturbing a hi
dden hive in a bush while trimming foliage around the premis
es. He immediately ran from the area, but had already sustai
ned several stings on his arms and neck. Co-workers administ
ered ice packs and called emergency services. Though he did 
not suffer a severe allergic reaction, the incident resulted
 in swelling and required a precautionary visit to urgent ca
re. The groundskeeping schedule was adjusted, and a pest con
```

Select model: Logistic Regression

The predicted category is: 1 VIOLENCE AND OTHER INJURIES BY PERSONS OR ANIMALS

Figure 8: Interface

11 Conclusion

The initial work requirements have been met, having identified two classification models with nearly identical performance. These models allow, given a textual report on a workplace accident in the construction sector, the classification of the triggering cause of the incident. The models in question, along with their test set results, are:

Model	**Accuracy**	**Weighted F1-Score**
SVM	0.90	0.90
Logistic Regression	0.91	0.91

Table 5: Performance comparison between SVM and Logistic Regression models

The analysis showed that the WCTF-IDF technique outperformed word embedding, effectively addressing the issue of minority classes being misclassified due to the presence of dominant class terms. As a result, WCTF-IDF emerges as a technique to be taken into consideration when dealing with highly imbalanced classification problems.

The results obtained were also comparable to those previously presented in the work from which the dataset originates.

Therefore, in a real-world context, our two models can be effectively used for classifying the causes of accident reports in the construction sector.

12 Bibliography

References

[1] Cheng, M. Y., Kusoemo, D., & Gosno, R. A. (2020). Text mining-based construction site accident classification using hybrid supervised machine learning. *Automation in Construction*, 118, 103265.

[2] Qiao, J., Wang, C., Guan, S., & Liu, S. (2022). Construction-accident narrative classification using shallow and deep learning. *Journal of Construction Engineering and Management*, 148(9).

[3] Deepwiz AI. (2023). How to correctly use TF-IDF with imbalanced data. Retrieved from `https://www.deepwizai.com/projects/how-to-correctly-use-tf-idf-with-imbalanced-data`