Master's Degree Program in

Data Science and Advanced Analytics

Specialization in Data Science

Machine Learning Project Report

To Grant or Not to Grant: Deciding on Compensation Benefits

Group 46:

Afonso Ascensão, 20240684; Duarte Marques, 20240522; Joana Esteves, 20240746; Rita Serra, 20240515; Rodrigo Luís, 20240742

NOVA Information Management School

NOVA University of Lisbon

2024/2025

Abstract

The New York Workers' Compensation Board (WCB) ensures benefits to workers, volunteers and civil defense personnel injured on the job. The WCB regularly reviews over 5 million claims since 2000, and partnered with Nova IMS to develop an algorithmic, data driven model to accelerate claim processing while increasing accuracy.

The goal of this project was to build a solid machine learning model capable of forecasting claim outcomes and dealing with data quality problems and extremely unbalanced target classes. Some of the main influential factors of claims are discussed through exploratory analysis. Preprocessing steps involved handling missing values, outliers and feature engineering to increase data quality. Feature selection was performed by ranking variables according to the combined statistical tests and their corresponding algorithmic ranks.

Several models were developed, including ensemble, instance-based learning and neural networks algorithms. The hyperparameter tuning of these models made use of Grid Search and further adjusted to reduce overfitting, while the final assessment figured a stratified K-fold Cross Validation with 5 folds to fairly compare the models.

XGBoost was selected as the final model for its efficiency and capacity to capture complex patterns in the data. SMOTE was selectively applied in the final model to address class imbalance and to improve predictions for underrepresented classes. A secondary model predicting "Agreement Reached"—a critical feature that significantly enhanced the primary model's performance—was also developed.

The final solution achieved a macro F1-score of 0.402, with consistent performance across validation folds. This solution automates the claim assessments thereby reducing the manual review, ensuring fastest possible decision making. It allows the WCB to work most effectively on the more complex cases and to allocate resources more efficiently.

Index

List of Figures

List of Tables

Introduction

This report outlines the process of developing our project, which aimed to create a multiclass classification model. The project encompasses all steps of the machine learning life cycle, from data exploration and preprocessing to model selection, evaluation, and optimization.

The data exploration section aims to understand the nature of the dataset at hand, identify characteristics that must be taken into account in the developing of the model, and inconsistencies present in the data. The subsequent preprocessing steps include outlier handling, imputations, reduction of categorical feature cardinality with encoding, and standardization. Additionally, new features from the feature engineering process are introduced, explaining the transformations applied to derive them. The preprocessing steps are divided into dataset-level transformations and post-train-test split procedures to ensure data integrity and avoid leakage.

The Feature Selection Strategy section outlines the steps taken to ensure the most relevant features are used in the models. This part of the report explains the division into unsupervised and supervised methods. It discusses the filter methods used, such as statistical tests, applied to identify redundant or irrelevant features based on their relationship with the target variable. Additionally, it introduces embedded methods, which make use of algorithm-specific importance rankings for a final selection.

Subsequently, the report outlines the model selection process, including the hyperparameter tuning strategy and the algorithms selected for the models. The assessment strategy justifies the use of Cross-Validation, the macro F1-Score metric, and complementary evaluation criteria. This chapter also addresses the pipeline employed to guarantee consistent preprocessing during model evaluation. The final model, with the algorithm “XGBoost”, is introduced along with a dedicated chapter on optimization efforts and the rationale behind these decisions. This includes a comparison of performance across train, validation, test sets, and cross-validation folds to evaluate alignment between test and cross-validation results.

The final chapter explains the development of a secondary model to predict the binary feature "Agreement Reached," enabling its inclusion as a feature in the primary model. The process covers hyperparameter tuning, feature selection, and assessment of the final model.

Throughout this project, several key questions emerged, which are explored in this report and addressed in the conclusion. These include the dataset's data quality issues, leading to decisions to minimize information loss and enhance the original information with new features. Furthermore, the severe target class imbalance necessitated an evaluation of techniques to mitigate its impact. The feature selection strategy raised concerns about generalization and data leakage, while model assessment required a fair approach to compare models under consistent circumstances and define the appropriate evaluation metric. Lastly, decisions about the final model and optimization strategies were made after consideration of possibilities, as the selection process proved more challenging than initially anticipated.

The implementation of a pipeline and the development of various models resulted in the generation of specific files from certain notebooks, which were essential for the proper functioning of other notebooks. To clarify these dependencies and demonstrate the flow of datasets and files across notebooks, we created a flowchart, as shown in Figure 3. Additionally, we prepared a comprehensive README file that outlines the correct sequence for running the notebooks.

1. Exploration and Preprocessing

2.1 Data Exploration

2.1.1 Detection of Data Inconsistencies

In the preliminary analysis, we removed rows with a 9-digit '*Claim Identifier*' lacking other variables information. After this process, 574026 records and 29 columns remained. Most of the features had a low percentage of missing values, likely due to data collection issues. However, variables such as '*C-2 Date*,' '*C-3 Date*,' '*First Hearing Date*,' and '*IME-4 Count*' had over 65% missing data, which may be attributed to unfiled forms or missed hearings. The '*OIICS Nature of Injury Description*' variable was removed due to 100% missing values. Additionally, 1407 records with inconsistent '*Assembly Date*' sequences were identified. No duplicates were found in the dataset.

2.1.2 Statistical Summary and Data Visualizations

Below is an overview of key insights derived from the descriptive statistics and visual analytics for both feature types:

Numeric Variables:

- **Age at Injury**: Right-skewed distribution, with values that suggest potential data errors and the existence of outliers (Figure 4). 'Age at Injury' and 'Birth Year' are highly correlated (Figure 5), sharing 108 unique values. However, 'Age at Injury' appears to have little impact on claim outcomes (Figure 6).
- **Average Weekly Wage:** High mean and standard deviation, indicating anomalies, with 61% zeros and extreme outliers raise data quality concerns.
- **Birth Year:** The minimum value of 0 suggests errors in the data, and nearly 5% of the values are missing. Given its high correlation with 'Age at Injury,' we decided to remove this variable.
- **IME-4 Count:** The distribution is right-skewed, with a concentration of lower values but noticeable outliers (Figure 4). This variable seems to strongly influence the target.
- **Number of Dependents:** The uniform distribution raises concerns about data collection (Figure 4). This variable does not appear to have much impact on the target.

Categorical Variables*:*

- **Alternative Dispute Resolution**: Few cases reach alternative resolution, often with no compensation.
- **Attorney/Representative:** The majority lacked an attorney or representative, though having one increases the likelihood of compensation.
- **District Name** and **Medical Fee Region:** Both display similar target class proportions across their categories, suggesting weak predictive power individually.
- **Zip Code**: Due to its high correlation with 'Medical Fee Region' and other location variables, along with inconsistencies and 5% missing data, we excluded this variable (Figure 7).
- **Gender:** Predominantly male, with low correlation to the target.
- **COVID-19 Indicator:** Most cases are unrelated to COVID-19. It is highly correlated with the three *'WCIO'* codes (Figure 8), but their distinct relationships with the target justify retaining them to evaluate their impact on the final model.

- **Carrier Type:** Certain categories dominate, with two 'Carrier Type' values significantly affecting outcomes. Higher salaries are less likely to reach an agreement, while age has no impact. Private carriers have more agreements, though the overall proportion remains low.
- **Agreement Reached**: Most cases result in no agreement. This variable strongly influences the target, with significant differences impacting its value.
- **WCIO.. Codes and WCIO.. Descriptions:** The variables related to "codes" show perfect correlation with their respective "descriptions" (Cramér's V = 1), allowing us to remove all *'WCIO (...) Description'* variables.
- **Claim Injury Type**: The target variable has a heavy class imbalance.

2.2 Data Preprocessing

The data preprocessing stage was divided into two categories, dataset level transformations and preprocessing steps that had to be implemented after the train-test split. The first category involved correction of data types, inconsistencies and part of feature engineering needed for missing values imputations. The remaining steps were integrated in a pipeline as custom transformers in an effort to avoid data leakage during Grid Search and Cross-Validation processes.

2.2.1 Data Types

As identified in the EDA, some features data types are incorrect, so we corrected this as follows: '*Age At Injury*', '*Birth Year*', *'IME-4 Count*' and '*Number of dependents'* were defined as integers because they are discrete variables; features that represent dates were changed to Datetime for proper date-related analysis; variables that were related to an identification code were set to Object because they have a categoric nature; and lastly '*Alternative Dispute Resolution*', '*Agreement Reached*', '*Attorney/Representative*', '*Agreement Reached*' and '*Covid-19 Indicator*' were set to Boolean as they represents a binary outcome.

2.2.2 Inconsistencies

Previously, we identified several inconsistencies in the data and addressed them as follows:

- '*Assembly Date*' earlier than '*Accident Date*': In these cases, the *'Accident Date'* was adjusted to match the *'Assembly Date'*, solving the inconsistency without discarding the affected rows.
- '*Average Weekly Wage*', *'Birth Year*', and *'Age at Injury'*: All zero values were replaced with NaN to facilitate subsequent corrections.
- '*Gender*' and *'Alternative Dispute Resolution'*: The values 'U' (in '*Gender*') and 'X' (in '*Alternative Dispute Resolution*') were replaced with NaN to streamline future preprocessing.

2.2.3 Handling Outliers and Missing Values

To ensure model robustness and minimize outlier influence, we applied the Interquartile Range (IQR) and percentile methods to key variables. For *'Age at Injury'*, we capped excessively high values using the IQR method to reflect a realistic worker profile, while for *'Accident Year'*, we established a lower limit for outdated records. We set the upper limit for *'IME-4 Count'* at the 99th percentile and capped *'Average Weekly Wage'* at the 95th percentile to mitigate the impact of exceptionally high salaries.

We filled the missing values in *'Average Weekly Wage'* with the industry-specific median to reflect each industry's wage distribution and for *Age at Injury*, we imputed missing values with the median

age after recalculating the age from the available birth and accident years. For categorical variables with missing data, we used the mode to ensure consistency.

2.2.4 Feature Engineering

To improve the model's performance, we generated additional features by deriving new variables from the existing dataset. These features were designed to capture relevant patterns and enhance the model's predictive capability, by applying the following transformations: binarization, logarithmic scaling, temporal lags, conversion of dates to years, merging binary features, calculating ratios, and flagging extreme outliers. A summary of the newly created features is provided in the Table 1.

2.2.5 Encoding and Scaling

We began encoding by grouping high-cardinality features like '*WCIO Part of Body Code*', 'WCIO Nature of Injury Code', and '*WCIO Cause of Injury Code'* based on the project's meta data to reduce dimensionality. These, along with '*Carrier Type*', '*District Name*', and '*Medical Fee Region*', were encoded using One-Hot Encoding. For other high-cardinality features such as '*Industry Code*', '*Carrier Name*', and '*County of Injury*', we created 'another' category for values below 1% frequency and used Frequency Encoding to reduce dimensionality.

Despite the presence of outliers, we selected the *MinMaxScaler* to ensure that all variables had the same weight. Additionally, this method was chosen for its ability to accelerate the convergence of methods like logistic regression and for being a simple and efficient technique.

2. Multiclass Classification

3.1 Feature Selection

The feature selection process involved the development of a strategy that would align with our future intentions to apply Cross-Validation, requiring a distinction between supervised and unsupervised approaches. While the latest was conducted separately, the supervised feature selection was implemented in a pipeline as a dynamic process. This allowed us to avoid data leakage by transferring any target related analysis to be performed inside the cross-validation flow. The feature selection strategy is outlined in the fluxogram presented in Figure 1.

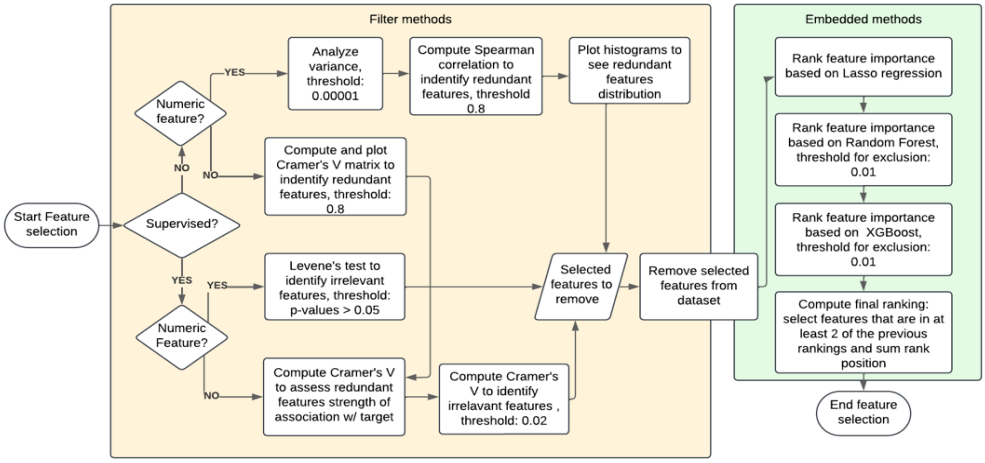

Figure 1 Feature Selection Process

Numerical Variables

Our selection process for numerical variables was based on criteria of statistical relevance and redundancy. Initially, we identified variables with near-zero variance, which indicated low discriminatory capacity. Subsequently, we employed Spearman's correlation to evaluate monotonic relationships, including non-linear ones. The variable *'Accident Year'* was excluded due to its high correlation (0.9) with *'Assembly Year',* low variance, and a higher initial number of missing values.

Variables such as *'Average Weekly Wage'* and *'IME-4 Count'* were replaced with their logarithmic transformations, as these transformations exhibited perfect correlation with the original variables, reduced skewness, and higher variance. Conversely, we discarded *'Wage Ratio'* due to its low variance and high correlation with other variables.

Finally, we applied Levene's test to assess the homogeneity of variances, excluding variables that did not show significant differences with respect to the target variable at a significance level of 0.05.

Categorical Variables

For categorical variables, we used the Cramér's V metric to identify redundant features and removed those that were less associated with the target variable compared to their counterparts.

Additionally, we applied a threshold of 0.02 to Cramér's V to exclude variables with weak associations to the target variable, ensuring the removal of features with minimal relevance.

Importance Assessment

After removing redundant and irrelevant variables, we created an importance ranking using three complementary methods: Lasso, Decision Trees, and XGBoost. We retained only variables that appeared in the rankings of at least two methods and summed their rank positions. Lasso contributed to dimensionality reduction by zeroing out irrelevant coefficients, Decision Trees captured non-linear relationships in an interpretable manner, and XGBoost highlighted complex patterns with high performance.

3.2. Models

We tested several models with different characteristics to compare the performance of their predictions. Our final analysis included: Instance-Based Learning – K-Nearest Neighbors; Artificial Neural Networks – MLP Classifier; Ensembled Algorithms – XGBoost, Random Forest and Stacking (combining predictions from XGBoost and MLP Classifier).

To determine the hyperparameters used in each model we selected a set of hyperparameters following documentation guidelines and performed a Grid Search to find the combination that originated the best macro F1-Score. The results of Grid Search often led to overfitting models, in such cases we made adjustments to reduce overfitting and optimize computational force.

All algorithms had difficulties predicting class 6 and 7 due to lack of observations and the Macro F1-Score for validation tended to decrease the hyperparameters were adjusted to reduce overfitting. To dela with class imbalance we used sample weights for the algorithms that supported this option, namely XGBoost and Random Forest to improve results for classes with a low number of observations. Oversampling techniques were briefly explored but they often lead to overfitting and the originated

features from feature selection when it was applied to oversampled datasets seemed to result less relevant features. The results of the models in this phase can be consulted on Table 2.

3.3 Model Assessment and Selection

To estimate our models' performance, we implemented stratified K-fold cross-validation. This approach was used for both parameter tuning and model selection. We applied this variation of K-fold cross-validation due to the heavily unbalanced dataset, because stratification ensures that class proportions across individual subsets matched those in the training set. This was especially important for model assessment, as we wanted to fairly compare models under the same circumstances, including the same class distribution. Additionally, we shuffled the order of samples within each class to ensure the original order did not influence the folds' composition.

Cross-validation provides a more accurate estimate of a model's prediction error compared to a single train-test split, as it averages performance across independent train-test splits, according to [1]. This motivated our choice, but the number of iterations introduces a trade-off between accuracy and computational expense. Considering this limitation, we set k=5, meaning the final macro F1-Score to consider for model selection was the average performance across five folds.

To avoid data leakage, we developed a pipeline for preprocessing steps and supervised feature selection. This ensured that after each split in the cross-validation process, the pipeline fitting was done on the current training set, while only the transformations were applied to the current validation set, preventing leakage of validation information into preprocessing and feature selection processes. The flow of the Cross-Validation process and its integration with the pipeline is illustrated in the fluxogram shown in Figure 2.

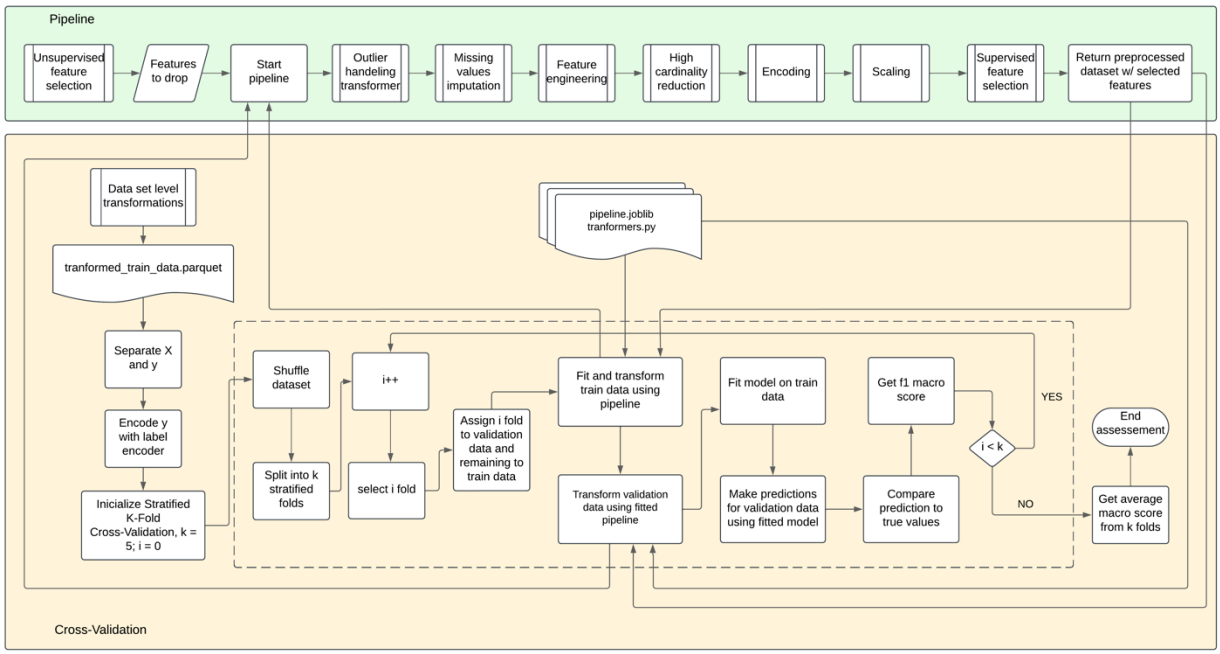

Figure 2 Integration with Cross-Validation and Pipeline

For the assessment of the models, we selected the macro F1-Score metric which evaluates the algorithm from a class standpoint, where a high macro F1-Score indicates the algorithm has a good performance on all the classes, as stated in [2]. Our goal was to develop a model that performed well across all classes, as we considered all of them equally important for the problem at hand. Our model selection strategy involved not only comparing the macro F1-Score but also analysing the individual F1-Score of each class. Given the unbalanced dataset, where minority classes were consistently harder to predict than majority classes across all models, we considered both the ability to predict all classes and the macro F1-Score when selecting the best model. Additionally, we considered the time each model took to complete, weighing whether the higher computational cost justified the higher score.

The final selected model was the XGBoost algorithm, as it provided a balanced choice across all our evaluation criteria. It performed faster than average, achieved a higher-than-average macro F1-Score, and made predictions for all classes. It is worth noting that even for this model, the performance on the minority classes was very poor. However, it was still the only model able to make predictions for these, suggesting that it may capture more patterns related to these classes than the others. (Table 3)

3.4 Model Optimization

After the selection of the final model, we made several efforts to improve its performance, and our first approach was to tailor parts of the preprocessing process. As stated in [3] encoding high categorical variables into numeric values can greatly reduce computational burden on tree-based models, additionally we believed category frequencies might relate to the assigned Claim Injury Type. Therefore, we modified the pipeline to encode all categorical features using a frequency encoder. Additionally, we adjusted the feature selection strategy to focus solely on XGBoost's feature importance rankings, prioritizing features more meaningful for this algorithm instead of a general feature selection process. However, these adjustments did not improve XGBoost's performance, and we reverted to the initial preprocessing and feature selection, as we suspected that key features for predicting minority classes were in the original feature set.

In a second attempt, we addressed the poor F1 scores for minority classes, which appeared to persist despite using a sample weighting strategy to handle class imbalance. To mitigate this, we implemented SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic observations, adding more data to classes with the lowest F1 scores, where the harder a class was to predict the more synthetic observations we generated. This approach improved overall performance and resulted in more balanced F1 scores across classes, although predictions for class 6 remained poor. While the macro F1 scores were consistent across the training, validation, and cross-validation folds, which lead us to believe the model generalizes well, the score dropped slightly on the test data. This decline, as well as the suboptimal F1 scores for certain classes, indicates the need for further refinements to improve the model's ability to generalize and handle minority class predictions. (Table 4)

3. Open-Ended Section

In an effort to enhance the basic model, we decided to develop a complementary model to predict the test data outcomes for the binary variable '*Agreement Reached*', enabling its use in the main model to predict the main variable. The same pipeline from the main model was also used for the secondary model, so the preprocessing and feature selection processes are largely similar, although the different target variable will lead to the selection of different features for prediction.

4.1 Models to test

To address the problem at hand, we selected two algorithms:

- **XGBoost Classifier**: This algorithm is very fast when compared to other models and incorporates regularization techniques to help mitigate overfitting.
- **KNN Classifier**: It is a straightforward algorithm that is easy to interpret and requires only a limited number of hyperparameters.

A third option was introduced subsequently: given the significant class imbalance in the dataset, we implemented oversampling, specifically using SMOTE (Synthetic Minority Over-sampling Technique).

4.1.1 XGBoost Classifier

In this model we tested several values for two key parameters: '*subsample*' (aimed at reducing overfitting) and '*scale_pos_weight*' (designed to address highly imbalanced data), starting with a "trial and error" approach to identify the ranges for which these parameters yielded the best results.

Subsequently, we conducted two separate Grid Searches, which were divided into two parts due to the exponential computational cost of the operation. These grid searches resulted in values of 0.95 for '*subsample*' and 3.9 for '*scale_pos_weight*'. Then, we looked for the values for the number of estimators, learning rate, tree depth and gamma ("*minimum loss reduction required to make a further partition on a leaf node of the tree*"), resulting in 2900 estimators, 0.01 learning rate, tree maximum depth of 9 and gamma 1.35.

However, the results conflicted with what we had learned in class, specifically the relationship between the number of estimators and the learning rate (i.e., a higher number of estimators should correspond to a lower learning rate). To solve this, we experimented with smaller learning rates while maintaining 2900 estimators, aiming to improve the macro F1-Score.

4.1.2 KNN Classifier

The kNN algorithm, not only was significantly slower, but the macro F1-Sore for validation was also underwhelming. Due to time constraints, we were unable to perform Grid Search for this algorithm, using instead the best performing value for *n_neighbors* (=13) and the "*kd-tree*" algorithm to optimize execution speed.

4.1.3 XGBoost Classifier with SMOTE

For the oversampled dataset, we used the same parameters as the imbalanced dataset, except for *scale_pos_weight*, as the dataset was no longer as imbalanced. The oversampling ratio that produced the best results was 0.2 (indicating a 0.2 ratio between the majority and minority class). Lower values resulted in worse performance, while higher values led to overfitting.

The results obtained from the models described above are presented in the Table 5. Based on these results, we selected XGBoost for the final model due to superior overall performance in terms of macro F1-Score a reduced overfitting compared to the other two models. Using this model, we were able to create a new feature that proved to be highly significant for the final model. During feature selection, '*Agreement Reached*' was among the top variables chosen.

Conclusion

During our EDA we noticed that the dataset had several issues and complex relationships. There were several features with missing values which resulted in some features exclusion and values imputations from central tendencies that could diverge from true values. Bivariate analysis highlighted some relationships between the features and the target, but no feature showed particularly strong correlation. The strong class imbalance in the target also raised questions about the correct prediction of minority classes. Feature engineering included a variety of transformations aimed at capturing more complex patterns, which succeeded as several of these features were retained after the feature selection procedure. Categorical cardinality reduction and encoding, allowed us to include categorical features in our models. We made use of one-hot and frequency encoding as complementary options based on the number of categories which resulted in a generalized approach for all models.

The feature selection process carefully addressed data leakage concerns, leading to the division of this process into supervised and unsupervised approaches. We combined statistical metrics (Levene's test, Spearman's correlation and Cramér's V) and three algorithm-specific importance rankings leveraging their complementary results for determining feature importance which allowed us to integrate the outputs into a final ranking. The feature selection strategy effectively removed irrelevant and redundant variables while ensuring a generalized selection of the most important features for all models.

Model testing included ensemble methods, namely boosting (XGBoost), bagging (Random Forest) and stacking. Additionally, we added diversity with instance-based learning (k-NN), and neural networks (MLP). In order to tune the hyperparameters of our models we implemented Grid Search to select the best combinations based on macro F1-score. This was initially assumed sufficient for optimization but frequently resulted in overfitting models that followed parameter adjustments based on trial and error to obtain better results. Grid Search helped understand what hyperparameters values resulted in a higher score, but it did not hold most of our final solutions. For the assessment of the models a stratified k-fold cross-validations was selected based on the nature of the dataset and to ensure reliable score. To avoid data leakage, we implemented a pipeline, which secured preprocessing and feature selection integrity and increased confidence in evaluation results.

Given the dataset's scenario, we suspected that more advanced models like XGBoost and MLP Classifier would perform better, due to their ability to detect more complex relationships. This was confirmed by the assessment scores, which lead us to develop a stacking model between these two algorithms. However, the slight performance improvement did not justify the computational effort, and some classes received no predictions. Therefore, XGBoost was ultimately selected as the more balanced choice, offering above average results, fast performance, and being the only model capable of predicting class 6.

To address target class imbalance, we initially believed sample weight could be a sufficient solution, but our efforts using this tool indicated otherwise. SMOTE with an equal number of observations was tested but often led to overfitting, and the feature selection process after applying SMOTE appeared to prioritize less relevant features. Therefore, we opted to apply SMOTE selectively to the optimized model, generating synthetic observations based on the F1-score of each class while retaining the initially selected features. This approach partially succeeded, improving the overall score, but it still fell short of achieving good results for all classes.

To further enhance performance, we developed a secondary model to predict the binary feature 'Agreement Reached' for the test dataset, enabling its inclusion in the main models. This feature consistently ranked among the top in our feature selection process, motivating the implementation of the secondary model. We believe that incorporating this feature benefited the primary model's performance and that this strategy holds potential for positively impacting future models.

The final model achieved a consistent macro F1-score across cross-validation folds, averaging 0.402. However, the slight decline in the test set score suggests the model may not generalize as well as initially anticipated. This indicates a need for further refinements to enhance robustness and ability to handle unseen data.

Throughout the project, we encountered several challenges at different stages of development. One of the main challenges was the highly imbalanced target classes, which significantly impacted the ability to predict minority classes accurately, lowering macro F1-scores and resulting in no score consistence across classes. The decision to use Cross-Validation for model assessment and hyperparameter tuning raised concerns about data leakage and to address this, we implemented a pipeline. However, the implementation presented technical challenges that required careful resolution to maintain data integrity and the reliability of the evaluation results. The implementation of Grid Search presented significant computational costs, often requiring several hours to produce results, which reduced the feasibility of trial-and-error adjustments with this tool due to time constraints.

Future work could explore different preprocessing options, including more advanced missing values imputation and outlier handling techniques. Additionally, different oversampling options could be explored and further hyperparameter tuning could enhance model performance. The current pipeline, feature selection strategy and assessment process provide a foundation for future efforts on model optimization.

All the work developed has been published on GitHub, making the code and results accessible for consultation and reproduction. (https://github.com/joanaefsteves/ML_Project_Group46.git)

References

[1] Ozdemir, S. (2016). Principles of Data Science.

[2] Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for Multi-Class Classification: An Overview.

[3] Zhu, W., Qiu, R., & Fu, Y. (2024). Comparative Study of the Performance of Categorical Variable Encoders in Classification and Regression Tasks.

Bibliography

Control Overfitting. (n.d.). Retrieved from XGBoost Documentation: https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html#control-overfitting

Alice Zheng and Amanda Casari. (n.d.). *Feature Engineering for Machine Learning PRINCIPLES AND TECHNIQUES FOR DATA SCIENTISTS.*

Assembled Workers' Compensation Claims Beginning 2001. (n.d.). Retrieved from Data.ny.gov: https://data.ny.gov/Government-Finance/Assembled-Workers-Compensation-Claims-Beginning-20/jshw-gkgu/about_data

Brownlee, J. (n.d.). *Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models and Work Projects End-To-End .*

Chen, T. &. (n.d.). *XGBoost: A scalable tree boosting system.*

Handle Imbalanced Dataset. (n.d.). Retrieved from XGBoost Documentation: https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html#handle-imbalanced-dataset

He, H. &. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*.

Kearney, M. W. (2017). *Sage Encyclopedia of Communication Research Methods.* Sage Publications.

Levene Test Tutorial. (n.d.). Retrieved from DATAtab: https://datatab.net/tutorial/levene-test

Parameters for XGBoost. (n.d.). Retrieved from XGBoost Documentation: https://xgboost.readthedocs.io/en/stable/parameter.html

sklearn.ensemble.RandomForestClassifier. (n.d.). Retrieved from scikit-learn: https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html

sklearn.impute.KNNImputer. (n.d.). Retrieved from scikit-learn: https://scikit-learn.org/1.5/modules/generated/sklearn.impute.KNNImputer.html

sklearn.linear_model.LogisticRegression. (n.d.). Retrieved from scikit-learn: https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html

sklearn.neural_network.MLPClassifier. (n.d.). Retrieved from scikit-learn: https://scikit-learn.org/1.5/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

Annex I

The propose of Annex I is to provide further clarification on the use of methods implemented that were not discussed in class.

Levene's test:

Levene's test is a statistical procedure used to assess whether the variances of a variable are equal across different groups. Although there is no direct application of Levene's test in feature selection, understanding the variability of variables in relation to the target can provide valuable insights in the modeling process. If a variable shows significantly different variances across target categories, this may indicate that it has useful discriminative power for the model.

Cramer's V:

- Adequate in categorical-categorical variables associations with 2 or more categories.
- It is easier to interpret than the chi-squared statistic, which may suggest a relationship between 2 variables, while the Cramer's V indicates the strength of association between two variables.

XGBoost and GBM comparison:

- Speed: Due to parallelization, XGB can run ten times as fast as sklearn's GBM;
- Missing Data Handling: Despite not having NaNs in our final delivery, this point was one of the main reasons why we used XGB in the first handout.
- Reduced overfitting: Through regularization, XGB can generalize better than sklearn's GBM.

Annex II

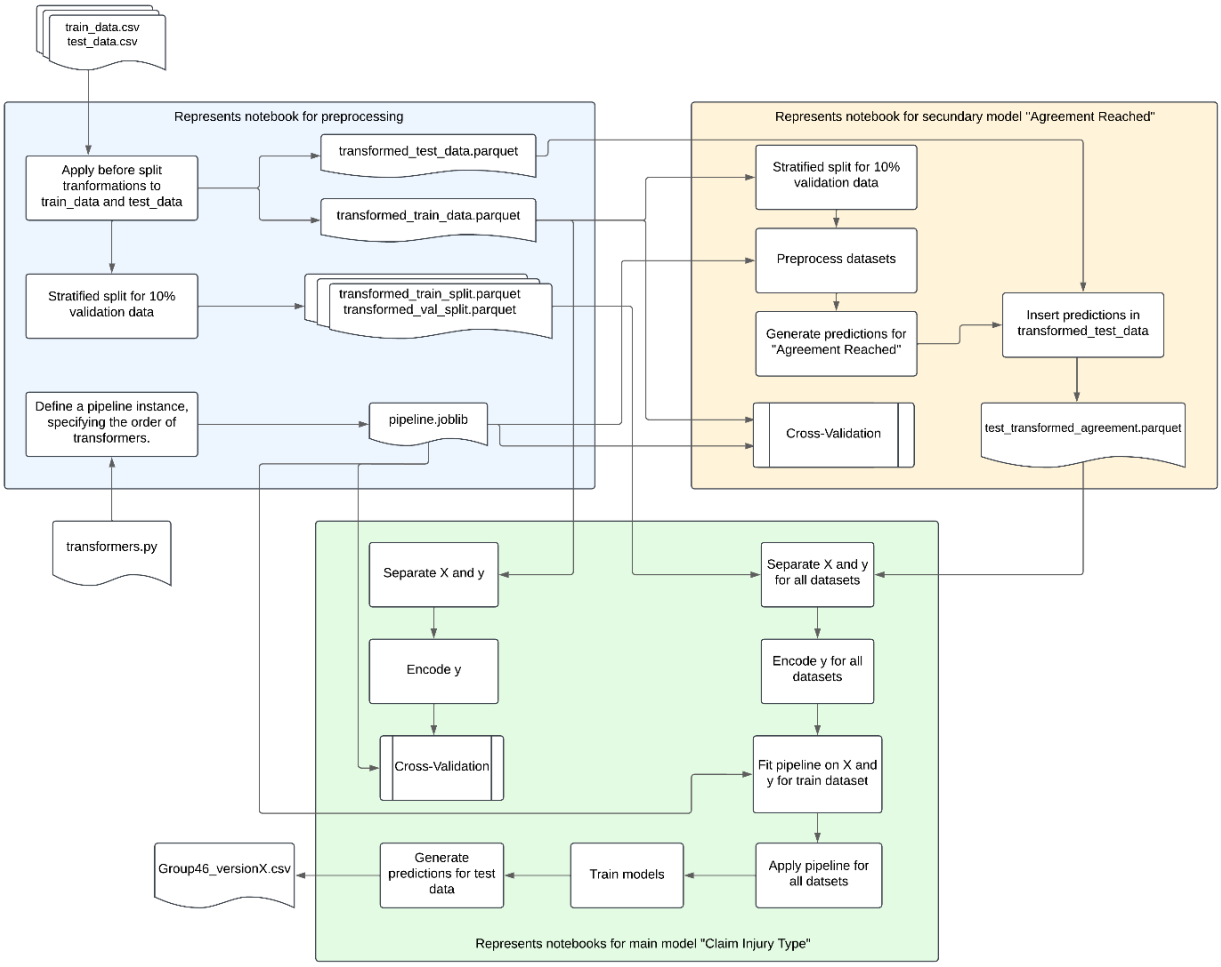

Figure 3 Dataset and File Dependency Flowchart

~

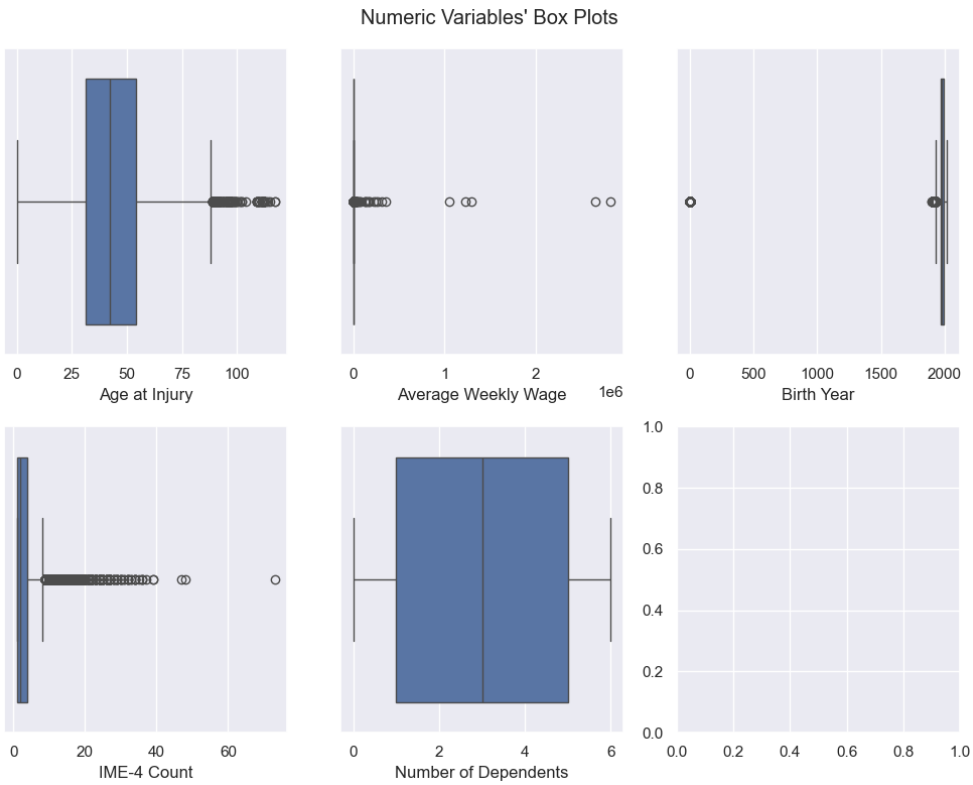

Figure 4 Numeric Variables' Box Plots

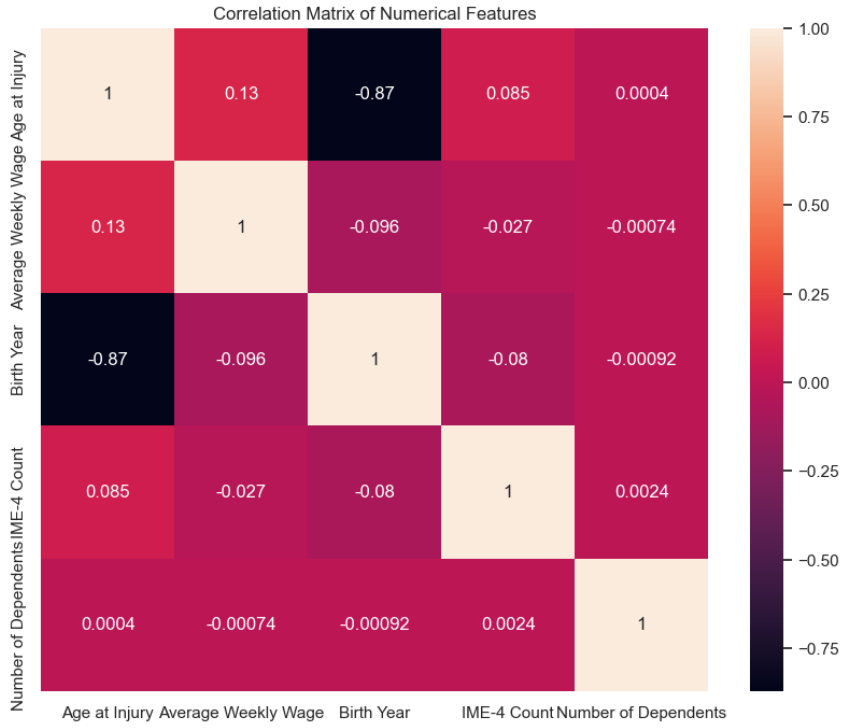

Figure 5 Spearman Correlation Matrix of Numerical Features

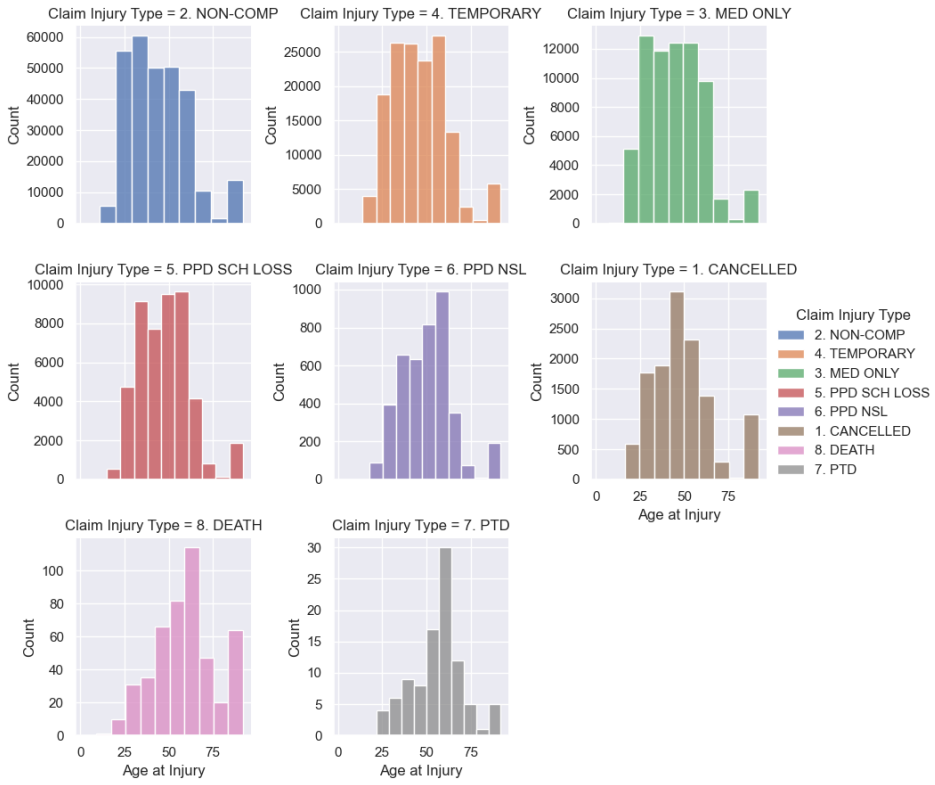

Figure 6 Age Distribution by Claim Injury Type

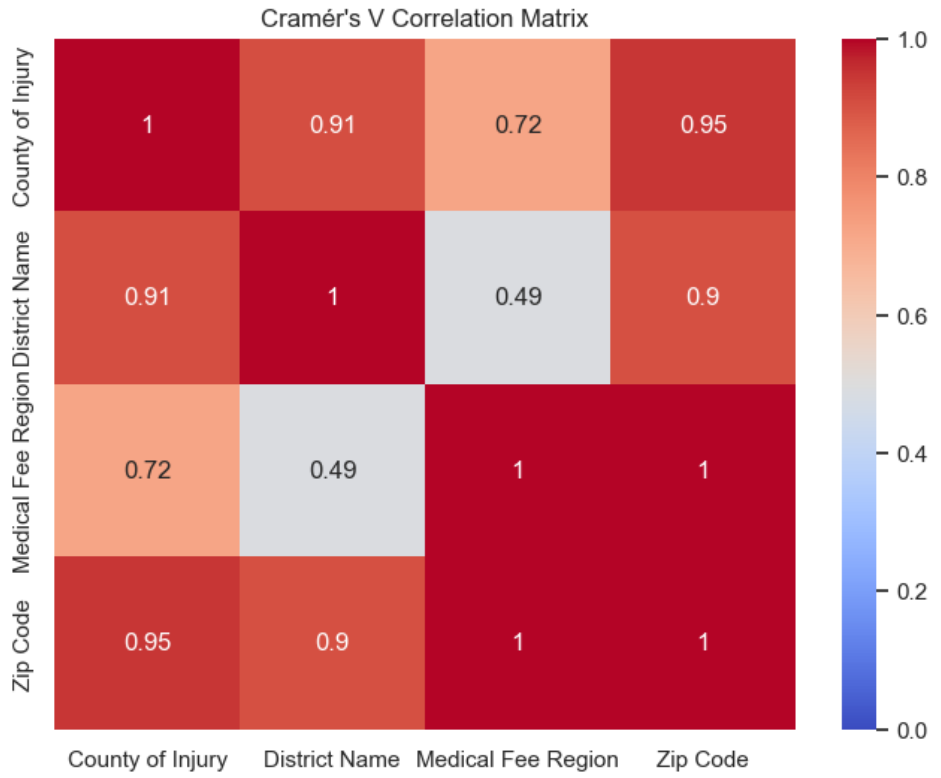

Figure 7 Cramér's V Matrix for some Categorical Variables

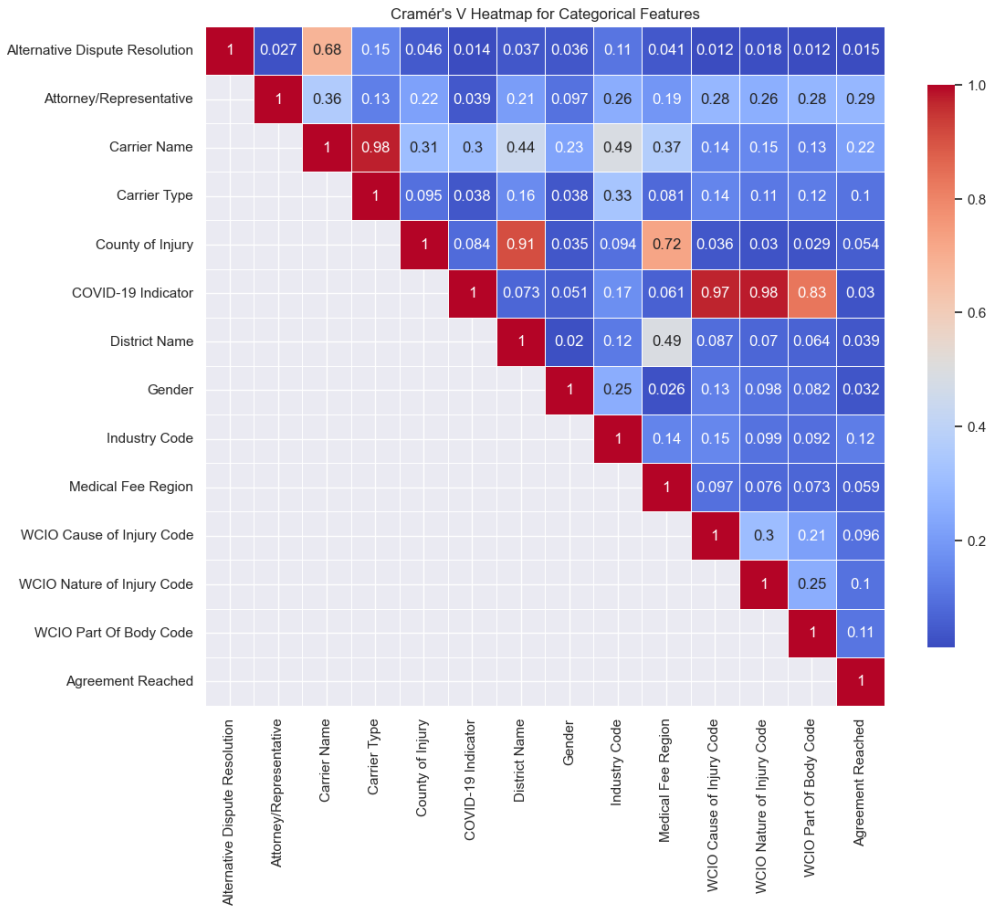

Figure 8 Cramér's V Heatmap for Categorical Features

Table 1 New Features Created

Transformation	Features	Description
Binary Features	Hearing Held	Indicates if a hearing was held or not
	C-2 Delivered	Indicates if this form was ever delivered
	C-2 Delivered on Time	Indicates whether the C-2 form was submitted within the established deadline (up to 10 days)
	C-3 Delivered	Indicates if this form was ever delivered
	C-3 Delivered on Time	Indicates if the form was delivered in the established time (up to 730 days)
Mathematical Transformations	Average Weekly Wage Log	The logarithmic transformation of the average weekly wage and the IME-4 count reduces the scale and asymmetry of the distribution
	IME-4 Count Log	
Temporal Lags	Time Accident to Assembly	Days between the accident date and assembly date
	Time Assembly to Hearing	Days between the assembly date and first hearing date
Dates to Year	Accident Year	The year when the accident occurred
	Assembly Year	The year when the claim was assembled
Union of Binary Features	At/rp OR altr dispute	Binary feature indicating whether at least one of the 'Alternative Dispute Resolution' or 'Attorney/Representative' events occurred
	Covid OR altr dispute	Binary feature indicating whether at least one of the 'Alternative Dispute Resolution' or 'COVID-19 Indicator' events occurred
	Covid OR At/rp	Binary variable indicating whether at least one of the 'COVID-19 Indicator' or 'Attorney/Representative' events occurred
Ratios	Wage Age Ratio	Ratio between the average weekly wage (after log) and age at injury
Extreme Outliers Flags	High Wage Flag	Indicates if the value of the average weekly wage is above the 95th percentile of the weekly wage
	High count IME-4 Flag	Marks values in the top 1% for the number of IME-4 forms delivered

Table 2 Hyperparameters of each algorithm and their respective Macro F1-scores

Algorithm	Hyperparameter's	Hyperparameter's used	Macro F1-score Train	Macro F1-score validation
XGBoost	n_estimators: [80, 120, 150], max_depth: [6, 8, 10, 12], Learning_rate: [0.01, 0.1, 0.2], Subsample: [0.4, 0.7, 0.9], Objective:'multi:softmax'	objective='multi:softmax, n_estimators = 150, max_depth = 10, learning_rate = 0.2, subsample = 0.7,	0.379	0.362
Random Forest	n_estimators: [150, 170, 180], max_samples: [0.5, 0.6], max_depth: [8, 9]	n_estimators = 180, max_samples = 0.4, max_depth = 9	0.409	0.367
KNN	model__n_neighbors: [10, 13, 15], model__weights: ["distance", "uniform"]	n_neighbors = 10, Algorithm = "kd_tree", Weights = "uniform"	0.303	0.300
MLP	model__hidden_layer_sizes: [(64, 32), (64, 64)], model__alpha: [0.001, 0.0001], model__learning_rate_init: [0.01, 0.001]	hidden_layer_sizes = (64,64), activation = 'relu', solver = 'adam', learning_rate_init=0.001, max_iter=1000, alpha=0.0001	0.417	0.405
Stacking: XGBoost and MLP Classifier		Default Logistic Regression parameters	0.459	0.416

Table 3 Scores obtained during cross-validation (CV)

Algorithm	Macro F1-Score CV	F1-Score train for each class
XGBoost	0.363574	0: 0.377342; 1: 0.842157; 2: 0.236901; 3: 0.629534; 4: 0.589511; 5: 0.125626; 6: 0.005391; 7: 0.090110
K-Nearest Neighbors	0.362593	0: 0.490403; 1: 0.855654; 2: 0.139678; 3: 0.696267; 4: 0.568835; 5: 0.009238; 6: 0.000000; 7: 0.178571
Random Forest	0.301929	0: 0.469228; 1: 0.824738; 2: 0.098501; 3: 0.348229; 4: 0.518360; 5: 0.092888; 6: 0.033647; 7: 0.038727
MLP Classifier	0.395329	0: 0.498224; 1: 0.894166; 2: 0.131731; 3: 0.778104; 4: 0.641419; 5: 0.018476; 6: 0.000000; 7: 0.285714
Stacking: XGBoost and MLP Classifier	0.409405	0: 0.498224; 1: 0.894166; 2: 0.131731; 3: 0.778104; 4: 0.641419; 5: 0.018476; 6: 0.000000; 7: 0.285714

Table 4 Model Performance Before and After the Application of SMOTE

Evaluation Metric	Scores
F1 scores across classes	**0:** 0.458; **1:** 0.884; **2:** 0.213; **3:** 0.692; **4:** 0.579; **5:** 0.137; **6:** 0.01; **7:** 0.262
Train macro F1 Score	0.419
Validation macro F1 Score	0.404
Macro F1 score across 5 folds	0.400; 0.407; 0.401; 0.404; 0.397
Cross-Validation average macro F1 Score	0.402
Test macro F1 Score	0.370

Table 5 Model performance in predicting 'Agreement Reached'

Algorithm	Precision- train	Precision-validation	F1 macro – train	F1 macro-validation
XGBoost	0.66	0.66	0.68	0.68
KNN	0.82	0.76	0.68	0.58
XGBoost with SMOTE	0.80	0.65	0.77	0.68