NOVA INFORMATION MANAGEMENT SCHOOL
Universidade Nova de Lisboa

MACHINE LEARNING PROJECT
Master in Data Science and Advanced Analytics

TO GRANT OR NOT TO GRANT

- Project Report -

GROUP 28

Afonso Silva, 20240495
Alice Viegas, 20240572
Beatriz Figueira, 20230479
Catarina Carneiro, 20240690
Pedro Trindade, 20240573

FALL/SPRING SEMESTER 2024-2025

TABLE OF CONTENTS

ABSTRACT

This project aims to develop a machine learning solution to predict the *Claim Injury Type* for the New York Workers' Compensation Board (WCB) using historical claims data. The dataset includes diverse features such as *Age at Injury, Medical Fee Region*, various accident details, and other metadata. The primary goal is to construct a robust multiclass classification model to automate and enhance the decision-making process for awarding compensation benefits.

Key steps in the project involve data preprocessing to address inconsistencies, categorical variable encoding, and feature selection. Additional features were engineered to improve model performance, ensuring more accurate predictions on the validation dataset.

Several machine learning algorithms were evaluated, including logistic regression, decision trees, and ensemble methods, to identify the best-performing model based on metrics such as accuracy, macro F1-score, recall, and precision. Techniques like Recursive Feature Elimination (RFE) and feature importance analysis were applied to identify the most influential variables.

Our solution identified LightGBM as the optimal model for automating claim classification, providing a reliable and efficient tool to reduce the WCB's manual workload. This demonstrates the power of machine learning in improving decision-making and streamlining large-scale administrative tasks.

Key Words: Machine Learning, Predictive Model, Classification problem, Random Forest, LightGBM

1. INTRODUCTION

The New York Workers' Compensation Board (WCB) manages claims for workplace injuries and other benefits. Processing the claims manually is time-consuming, so the WCB partnered with Nova IMS to create an automated model to predict *Claim Injury Type*, using data from 2020 to 2022, to streamline their workflow. For this we used a set of labelled data with all claims assembled between 2020 and 2022 that reached the WCB, provided by the company.

Our project has three main goals. The first is to build a classification model that predict what type of injury (*Claim Injury Type*) should be given to each claim. To do this, we were expected to compare different candidate models, evaluate their performance metrics, and finally decide on the most generalizable one.

The second objective is to further explore and optimize the model's performance by adjusting the previous pre-processing and feature selection processes as well as assessing imbalanced learning techniques. Our goal is to build a model with a good f1 score that runs efficiently and can generalize well.

The final, and third, objective of the project is to develop an open-ended segment meant for deriving complementary insights from the dataset. In this segment we were free to explore any idea we found relevant or interesting to improve our model. For this end, we decided to investigate if predicting a variable such as *Agreement Reached* (which does not exist in the testing set) and using this variable as a feature in our main model (which predicts *Claim Injury Type*) can improve its performance.

Since the previous delivery, we have fundamentally changed the project and its structure. These changes were made to address and integrate the valuable feedback provided by the teacher, ensuring the work is better aligned with the proposed objectives.

2. DATA EXPLORATION AND PREPROCESSING

2.1. DATASET DESCRIPTION

It is crucial to start by understanding what the variables provided mean, check the shape of the dataset and if we have missing and/or duplicated values. The project dataset was provided in two separate CSV files: one for training and one for testing. These datasets contain information about the worker's demographics, claim specifics and accident and injury details:

- **Training Dataset**: The training dataset consists of 593 471 rows and 33 columns, of which about 36,3% correspond to numerical variables (both integers and floats) and 63,6% to categorical variables (as objects). This dataset was used for exploratory data analysis, data cleaning, feature engineering, and model training.

- **Testing Dataset**: The testing dataset contains 387 975 rows and only 30 columns. It contains the same columns from the train dataset except for *Agreement Reached, Claim Injury Type* (the target variable) and *WCB Decision*. This dataset was reserved exclusively for model evaluation and was not used during exploratory data analysis to prevent data leakage.

During exploratory data analysis (EDA), it's important to check for duplicates, which are repeated rows of data. If not properly dealt with, duplicates can introduce bias and lead to overfitting in the model and distort statistical results. In our case, we verified that our data contains no duplicates neither in the training or test datasets.

2.2. DESCRIPTIVE STATISTICS

The train dataset reveals several important characteristics and potential data quality issues in the dataset. Looking at the numerical variables, the average *Age at Injury* is approximately 42 years, which seems fair, but the values range from 0 to 117, suggesting anomalies or wrong data entries. Additionally, some records indicate children with dependents and reporting workplace injuries, which supports the previous suggestion. Similarly, *Average Weekly Wage* includes extreme values, with a maximum of 2 828 079 and a minimum of 0, which is rather unlikely. *Birth Year* contains a lot of values as 0.0 which most likely indicate missing entries, while other columns like *Number of Dependents* appear consistent.

Some features exhibit high cardinality, which is memory and time inefficient, and will end up negatively impacting future modelling. All code variables derived from the WCIO website, along with *Industry Code, OIICS Nature of Injury Description* and *Agreement Reached*, are wrongly identified as numerical when they have categorical nature.

For categorical variables, ambiguous categories, such as "X" in the *Gender* column, may require cleaning or imputation to ensure accurate analysis. Additionally, *WCB Decision* has only one unique value ("Not Work Related"). As can be seen in Table 1, The column *OIICS Nature of Injury Description* contains only NaN values failing to provide any information to the model and features like *C-3 Date*, *IME-4 Count* and *First Hearing Date* show high percentage of missing values, reducing their usefulness. Regarding our target variable, approximately 49% of the observations belong to the majority class, highlighting a significant class imbalance in *Claim Injury Type* (Figure 1).

Column	Non-Null Count	Dtype
Accident Date	570337	object
Age at Injury	574026	float64
Alternative Dispute Resolution	574026	object
Assembly Date	593471	object
Attorney/Representative	574026	object
Average Weekly Wage	545375	float64
Birth Year	544948	float64
C-2 Date	559466	object
C-3 Date	187245	object
Carrier Name	574026	object
Carrier Type	574026	object
Claim Identifier	593471	int64
Claim Injury Type	574026	object
County of Injury	574026	object
COVID-19 Indicator	574026	object
District Name	574026	object
First Hearing Date	150798	object
Gender	574026	object
IME-4 Count	132803	float64
Industry Code	564068	float64
Industry Code Description	564068	object
Medical Fee Region	574026	object
OIICS Nature of Injury Description	0	float64
WCIO Cause of Injury Code	558386	float64
WCIO Cause of Injury Description	558386	object
WCIO Nature of Injury Code	558369	float64
WCIO Nature of Injury Description	558369	object
WCIO Part Of Body Code	556944	float64
WCIO Part Of Body Description	556944	object
Zip Code	545389	object
Agreement Reached	574026	float64
WCB Decision	574026	object
Number of Dependents	574026	float64

Table 1 - Missing Values

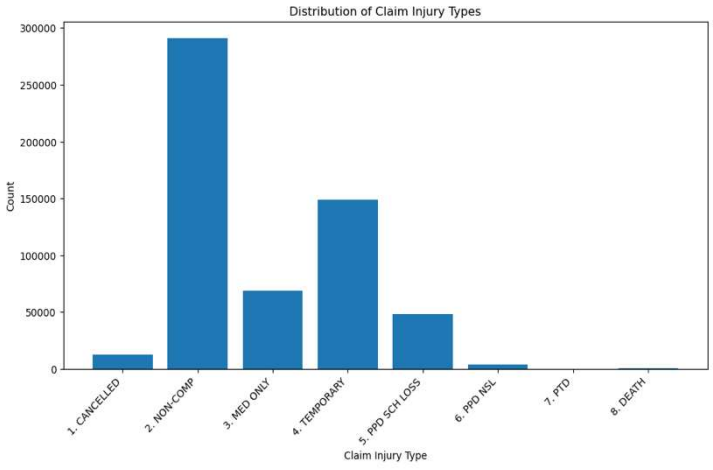

Figure 2 - Distribution of target variable

2.3. DATA CLEANING

Data cleaning refers to the process of correcting or removing errors, inconsistencies, and missing values in a dataset to improve its quality and usability for analysis. Additionally, made sure that all transformations were consistent in the training and test datasets.

2.3.1. COLUMNS AND ROWS DROPPED

The column *OIICS Nature of Injury Description* contains no information showing no variability, therefore, it can be safely dropped. Similarly, *WCD Decision* was dropped not only because it contains only one unique value, providing no utility for analysis but because it is not present in the test dataset and will not be used in the open-ended section, unlike *Agreement Reached*.

Due to their high cardinality, the columns *Zip Code* and *Carrier Name* were also dropped. This decision was driven by the fact that these features provide very specific information while other columns, such as *County of Injury* and *Carrier Type*, offer similar insights with significantly lower cardinality.

We observed that for all rows where the target was unknown, no information was provided in the other columns, except for *Assembly Date* and *Claim Identifier*. Since *Claim Identifier* is unique to each row, it does not contribute to generalizable patterns for the model. We also examined the distribution of rows with an unknown *Claim Injury Type* across the *Assembly Date* years. Since they were evenly distributed, we concluded that *Assembly Date* doesn't provide meaningful temporal information either. Also, rows where the *Accident Date* occurred after the *Assembly Date* were dropped, given the impossibility of such an event and the inability to determine which of the two dates was recorded incorrectly.

Maintaining these columns and rows would only add unnecessary complexity without meaningful improvement to the model's predictive power.

2.3.2. VALUE INCOHERENCES

In the columns *Birth Year* and *Age at Injury*, we replaced all values equal to 0.0 with NaN, as it is highly unlikely for a person to have a work injury at birth or to be born in legal year 0.

For the *Average Weekly Wage* column however, we chose to keep the zeros, recognizing that a zero wage is a valid possibility. This can occur in situations where the individual is a volunteer, unpaid worker (such as an intern), independent contractor, or unemployed at the time of the injury— for example, an injury that occurred during a job interview. These cases reflect real-world situations and should not be excluded.

2.3.3. CHANGES IN DTYPES

Data types wrongly identified as numerical in the previous section were changed to str type. Additionally, we made sure all date variables (*Accident Date, Assembly Date, C-2 Date, C-3 Date, First Hearing Date)* were transformed to *datetime*. Though *datetime* is a subclass of object, converting variables to this specialized type improves memory efficiency, performance, and enables type-specific operations. Plain *object* variables are treated as strings, limiting both performance and functionality.

2.3.4. IMPUTING MISSING VALUES WITH ACCIDENT DATE

By leveraging *Accident Date*, we were able to impute missing values in columns *Birth Year* and *Age at Injury*. Specifically, for *Birth Year*, we subtracted the *Age at Injury* from the *Accident Date* year, providing an accurate estimate. For *Age at Injury*, we reversed the process, subtracting the *Birth Year* from the *Accident Date* year.

2.4. PREPROCESSING

For this step, where we prepare raw data for modelling by cleaning, transforming, and organizing it to ensure its suitability for analysis and machine learning. First, we decided to set *Claim Identifier* as our dataset's index, given that at this stage it contained no duplicated values. The only duplicated value in this column was deleted in the previous section when excluding the rows where the target variable is missing. To standardize the categories into a simpler, machine-readable format, making it easier to process and analyse we mapped *Claim Injury Type* accordingly to the numeric prefix of each category (so from 1 to 8) and then applied this mapping to the target column.

At this stage, and before performing any actual preprocessing, we separated numerical and categorical features to apply the appropriate transformations specific to each type and split the data into training and test sets to prevent data leakage and ensure that the model is evaluated only on unseen data, remaining unbiased.

After consulting the WCIO website, we noticed that *Part of Body Code* -9 did not exist in their system and should therefore have been wrongly recorded. At this point we had 42 011 rows assigned to *Part of Body Code* -9 and all of them has the same *Part of Body Description*, "MULTIPLE". From this, we assumed these rows were supposed to have a *Part of Body Code* from the section "Multiple Body Parts" which includes codes 64, 65, 66, 90, 91 and 99 as stated on the website. We decided to substitute -9 codes with the most common code from "Multiple Body Parts", 90. However doing this step in data cleaning could introduce data leakage so it was dealt with in the next section after splitting the training dataset into train and validation subsets.

2.4.1. FEATURE ENGINEERING

In order to derive higher utility and efficiency from our remaining high cardinality categorical features, for each one, we created a simplified variable where their values were grouped into broader, more meaningful categories using mapping. Besides the below mentioned variables, other new ones were created from existing ones and are shown in Table 3 along with their data types (Dtypes) and descriptions.

High Cardinality Variable	Simplified New Variable	Dtype of New Variable	Description
Industry Code	Industry Code_Risk_Level And Industry Code_N_Entities	float	Groups Industry Codes into three categories based on risk of possible job-related injuries. The second shows the number of American entities related to each industry area codes.

WCIO Cause of Injury Code	WCIO Cause of Injury Code_Group	float	These variables were grouped based on the groups retrieved from the WCIO website.
WCIO Nature of Injury Code	WCIO Nature of Injury Code_Group	float	
WCIO Part Of Body Code	WCIO Part Of Body Code_Group	float	

Table 1 – New variables created from high dimensionality features

2.4.2. MISSING VALUES IMPUTATION

In the missing value imputation process, multiple strategies were employed to handle missing data effectively This combination of context-specific imputation methods ensures the data remains robust, minimizing the loss of information and preserving data patterns.

For the *County of Injury* column, "UNKNOWN" values were imputed with the most common county based on the corresponding *District Name*, ensuring the imputed values aligned with regional distributions. Similarly, "UK" values in the *Medical Fee Region* column were imputed using the most frequent region associated with each *County of Injury*. For the *IME-4 Count* column, missing values were filled with zero, as this likely represents that no IME-4 forms were received. Additionally, for numeric features such as *Accident Date Yea*r, missing values were imputed by summing *Age at Injury* and *Birth Year.* The remaining categorical columns had their missing values imputed using the mode, while the remaining continuous numeric columns were imputed using the median to prevent skewness caused by outliers. Some of the categorical variables missing values were stored as strings, so they had to be replaced by NaN type values and then imputed using the mode.

2.4.3. SCALING OF NUMERICAL VARIABLES

For scaling, numerical variables were standardized using the RobustScaler from the sklearn.preprocessing module. Robust scaling was specifically chosen because it scales the data based on interquartile range (IQR), making it highly robust to outliers. Unlike other scaling methods such as Min-Max scaling or standardization (z-score), RobustScaler does not require explicit prior handling or removal of outliers, as it minimizes their influence during the scaling process.

The scaler was first fitted to the numerical independent features of the training set to learn the scaling parameters and was, subsequently, applied to the validation and test set in order to maintain consistency and avoid data leakage. This approach ensures that numerical features with varying magnitudes are effectively scaled, while the impact of outliers is mitigated.

2.4.4. ENCODING OF CATEGORICAL VARIABLES

Regarding categorical variables, these were processed using label encoding to convert them into numerical representations suitable for machine learning models. Next, a roman-to-numeric mapping was applied to features like *Medical Fee Region* to convert Roman numeral categories into their respective numeric equivalents. Similarly, a binary mapping was created and applied to all binary columns. For features *County of Injury* and *District Name*, a mapping of counties and district names was created to transform textual categorical values into numeric codes. A similar process was applied to *Gender* and *Carrier Type*.

The mappings were applied across the train, validation, and test datasets to avoid discrepancies and ensure consistency. Finally, unique values for each categorical column were displayed to validate the encoding process.

Additionally, unnecessary description columns (*Industry Code Description* and WCIO description columns) were dropped, as they contained redundant information that was already present through other features.

3. MULTICLASS CLASSIFICATION

3.1. CLASS IMBALANCE

To address the class imbalance, present in our dataset, we employed a hybrid resampling technique. For this we first performed a variation of SMOTE that works specifically with categorical and continuous features, SMOTENC. Initially we identified the categorical columns in our dataset and applied SMOTENC to generate synthetic samples for the minority classes, ensuring a balanced class distribution. The sampling strategy was then set to over sampling the minority classes until they reach a certain threshold. The threshold was the number of instances in the third most popular target class, class 3. After over-sampling with SMOTENC, we applied Random Under-Sampling (RUS) to reduce the data size and mitigate noise caused by the excess of synthetic data. This combined approach was able to balance the class distribution while maintaining computational efficiency.

3.2. FEATURE SELECTION

To optimally select our final features before modeling, we combined filter, wrapper, and embedded methods. This approach balances efficiency (filter), precision in evaluating subsets (wrapper), and model-specific relevance (embedded), ensuring an optimal and well-informed feature set.

3.2.1. FILTER METHODS

Spearman correlation matrix: The Spearman correlation matrix for numeric features helps us draw conclusions about monotonic relationships and the strength of associations between variables. We consider as highly correlated the pairs of variables with a correlation above 0.7 (Figure 2).

G - test: The G-Test tests the association or independence between categorical variables. By analysing the test output, we identify variables that significantly influence the target and are thus valuable for predictions. This is illustrated by Figure 3.

Cramer's V: Cramér's V is a complementary approach to the G-Test, measuring the strength of association between categorical variables after testing them for dependency. In our case, we considered variables with scores above 0.3 as relevant and selected those for the model.

3.2.2. WRAPPER METHODS

RECURSIVE FEATURE ELIMINATION (RFE) WITH LIGHTGBM: By iteratively eliminating less important features and retraining the model, we evaluated the performance using the macro F1-score on both training and validation datasets. The results indicated that the optimal number of features was 16. However, the results for the test set showed no significant improvement in model performance after selecting more than 9 features, as the validation scores plateaued (Figure 4), so we changed the optimal solution to 9 features.

RECURSIVE FEATURE ELIMINATION (RFE) WITH LINEAR REGRESSION: When using LightGBM, we removed 7 features, retaining 9 for good baseline performance. This insight allows us to start the RFE with Logistic Regression

from 9 features onward, making it computationally more efficient. This time, the optimal number of features based on the training set performance was 10, but the number of features that prioritizes the model's ability to perform on unseen data is 4 (Figure 5), based on improvement per feature.

We chose to perform RFE with both algorithms given that Logistic Regression captures linear relationships, while LightGBM handles complex non-linear interactions, providing a well-rounded analysis of feature-target relationships.

3.2.3. EMBEDDED METHODS

LASSO WITH CROSS-VALIDATION: Lasso operates with numeric features, encouraging sparsity in the coefficients by shrinking less important feature weights to zero. Even though the coefficients of *Average Weekly Wage, Days_to_first_hearing, Days_between_c2_and_c3 and Days_to_assembly_date* were not zero, the magnitude of their coefficients is negligible compared to the other coefficients. For this reason, in summary table 5, they were flagged with a question mark instead of a "yes". In table 4, we can see the lasso coefficients for the features.

CATBOOST: CatBoost is a gradient boosting algorithm designed to handle categorical features efficiently, requiring minimal preprocessing. We used the average contribution score as the decision threshold to retain only the most impactful features (Figure 6).

3.2.4. FINAL DECISION

After careful consideration of all the methods employed, the following ones passed with majority vote and were ultimately selected for the modeling phase. In the annex is the full table with all variables and their results under each methodology (table 6).

Method Type	***Filter***			***Wrapper***		***Embedded***		***Final***
Feature	*Spearman*	*G-Test*	*Cramér's V*	*RFE LightGBM*	*RFE LR*	*Lasso*	*CatBoost*	
Age at Injury	*?*	-	-	*Yes*	*No*	*Yes*	-	*Yes*
Attorney/Representative	-	*yes*	*Yes*	-	-	-	*Yes*	*Yes*
Average Weekly Wage	*?*	-	-	*Yes*	*Yes*	*?*	-	*Yes*
WCIO Cause of Injury Code	-	*Yes*	*Yes*	-	-	-	*No*	*Yes*
WCIO Nature of Injury Code	-	*Yes*	*Yes*	-	-	-	*No*	*Yes*
WCIO Part Of Body Code	-	*Yes*	*Yes*	-	-	-	*Yes*	*Yes*
WCIO Cause of Injury Code_Group	-	*Yes*	*Yes*	-	-	-	*Yes*	*Yes*
WCIO Part Of Body Code_Group	-	*Yes*	*Yes*	-	-	-	*Yes*	*Yes*
Missing_Count	*?*	-	-	*Yes*	*Yes*	*Yes*	-	*Yes*
had_a_hearing	-	*Yes*	*Yes*	-	-	-	*No*	*Yes*
has_IME_4_Count	-	*Yes*	*Yes*	-	-	-	*Yes*	*Yes*
Average Weekly Wage_is_null	-	*No*	*Yes*	-	-	-	*Yes*	*Yes*

Table 5 - Final selected features

Uncertainties associated with any decision were flagged with a question mark, and, in case of a tie, the feature was classified as "no" and excluded from the model.

Worth noting was the case of *IME-4 Count*: To avoid multicollinearity, and even though *IME-4 Count* did pass with majority vote, we chose to discard it from the model given its high correlation with Missing Count (that passes with one vote more than *IME-4 Count*).

3.3. PERFORMANCE COMPARISON BETWEEN MODELS

For the modeling part of the project we chose, as requested, five candidate algorithms: Logistic Regression, Decision Tree, Gaussian Naive Bayes Classifier, K-Nearest Neighbors Classifier (KNN) and Linear Support Vector Classifier (SVC). The performance of the algorithms was assessed using a performance table with several evaluation metrics, such as Precision, Recall, macro F1-Score, Accuracy and Overfitting score. The Overfitting score was calculated by the difference in AUC score for the training and validation sets. A model with a high AUC score in training and a low AUC score in validation is considered overfitting in this approach. This performance table also performed One Hot Encoding by selecting only binary features, for some models that can be biased when handling categorical variables in the Label Encoding format. The models that needed this transformation were the Logistic Regression, K Nearest Neighbors (KNN) and Linear Support Vector Classifier (Linear SVC). The other models were dealt with using the Label Encoding technique for categorical features.

We wanted to demonstrate that key imbalanced learning techniques such as Ensemble, Re-sampling, Model Weights effectively improve the prediction of minority target classes. For this, we started by evaluating the five baseline models on our dataset without applying any imbalanced learning techniques (Table 6). These scores served as a baseline to understand how much these techniques would improve the predictions.

Ensembles: We tested RandomForest (bagging algorithm), AdaBoost and LightGBM (both boosting algorithms) and verified an overall improvement on the performance scores, especially LightGBM (Table 7). Stacking algorithms were not considered given their computationally intensive nature.

Re-sampling: For A hybrid approach we reused the combination of oversampling with SMOTENC and under-sampling with RandomUnderSampler used before prior to feature selection. Table 8 shows the results for this test. Then we also tested only using oversampling and only using undersampling (Tables 9 and 10).

Model Weights: we tested class weighting on our models by applying parameter class_weight='balanced' in Random Forest and AdaBoost's base estimator, and parameter is_unbalance = True in LightGBM. This ensured the models accounted for minority classes. (Table 11).

We ended up reaching the conclusion that the best approach would be to combine ensembles with hybrid re-sampling. This meant reusing the combination of SMOTENC oversampling and RandomUnderSampler under-sampling used before feature selection. The Light GBM model proved to be the one with the highest overall performance:

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
LightGBM	0.383455	0.533994	0.412904	0.714323	0.055187

Table 12 – Final performance scores table

4. OPEN-ENDED SECTION

In the open-ended section, we investigate if predicting *Agreement Reached* (which does not exist in the testing set) and using it as a feature in our main model (which predicts *Claim Injury Type*) can improve its performance. This was an interesting hypothesis given that there was a statistical test in feature selection that considered this variable relevant for predicting our target variable.

First, we selected the most relevant features for predicting Agreement Reached. Some of these features were IME4-4 Count, Accident Date_Year and Carrier Type. Then we tested some models to see which one would perform best at predicting Agreement Reached in the validation dataset with the selected features. After deciding in favor of Random Forest due to its superior F1-score, we went to check if using this feature on the validation dataset would help the model get a better F1-score.

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
LightGBM	0.398260	0.540788	0.427305	0.732369	0.052952

Table 13 – Open-ended section conclusions

The results were very good given that the f1-score increased more than 0.01 just for adding a binary feature. The next step was to train the Random Forest Classifier and predict Agreement Reached on the test dataset. These predictions were stored in a variable and then added to the test dataset as a column. Finally, we trained a Light GBM model with all the selected features including Agreement Reached in the hybrid resampled training data. The predictions were reverse encoded to the original Claim Injury Type classes and joined together in a data frame with their respective Claim Identifier. The data frame was exported as a csv file and named submission_file.csv.

5. CONCLUSION

In this project we addressed the main objectives of developing a robust multiclass classification model aimed to predict the Claim Injury Type for the WCB, improving its performance, and investigation of open-ended strategies in an attempt to see if more could be learned from the dataset.

We prepared a clean and structured dataset for modeling by preprocessing, feature engineering and handling of class imbalance using SMOTENC and Random Under-Sampling techniques. Different feature selection techniques were applied in order to improve the robustness of the model. We compared several machine learning algorithms; of these, LightGBM had the best performance, as it was expected, because of its ability to handle complex interactions in data.

In the open-ended section, we investigated the effect of predicting an intermediate variable, Agreement Reached and incorporating it in the main model. Including this feature led to improvements on the model scores, which supported our hypothesis that intermediate predictions could improve the model accuracy.

Despite the successes, there were some limitations: The dataset was very imbalanced, which, even using hybrid resampling techniques, may still have caused bias. High cardinality categorical variables were filtered out to prevent overfitting and speed up computation; however, their removal might have eliminated features that contained valuable information. Finally, relying on a predicted feature left the final model open to errors that may have been carried from this variable.

In future work, we could focus on tuning the parameters of each model as well as exploring other types of techniques aside from tree-based models. Also, it would be important to conduct a more robust evaluation pipeline like cross validation and exploring more encoding and scaling techniques.

In the final analysis, the project demonstrated the capability of machine learning to improve and automate the decision-making process regarding injury claim classification. The results presented should lay a strong foundation for continuing the development of predictive analysis in this context.

ANNEX 1 – TABLES

New Variable	*Dtype*	*Description*
Missing_Count	float	Indicates how many missing values per row
Under Age	int	Whether the subject had less than 18 years at the time of the accident (Binary)
Age_is_null	int	Whether Age at Injury is missing or not (Binary)
days_to_assembly_date	float	The difference, in days, between Accident Date and Assembly Date
days_between_c2_and_c3	float	The difference, in days, between C-3 Date and C-2 Date
days_to_first_hearing	float	The difference, in days, between First Hearing Date and Accident Date
Accident Date_Year	float	The year of Accident Date
Assembly Date_Year	float	The year of Assembly Date
C-2 Date_Year	float	The year of C-2 Date
C-3 Date_Year	float	The year of C-3 Date
First Hearing Date_Year"	float	The year of First Hearing Date
had_a_hearing	int	Whether there is record of a First Hearing Date for the subject or not (Binary)
has_IME_4_Count	int	Whether IME-4 Count of the subject is missing or not (Binary)
Industry Code_N_Entities	float	How many entities with the rows' Industry Code there are in the U.S. (Retrieved from the NAICS Website)
Average Weekly Wage_is_null	int	Whether Average Weekly Wage of the subject is missing or not

Table 2- Additional new variables

Feature	*Coefficient*	*Interpretation*

Age at Injury	0.535651	More important, strong positive relationship
IME-4 Count	0.080470	Important, but with less influence
Average Weekly Wage	0.006503	Very small contribution
Days_to_first_hearing	0.000764	Low relevance
Birth Year	- 0.000000	No impact → zero coefficient
Number of Dependents	0.000000	No impact → zero coefficient
Industry Code_N_Entities	- 0.000000	No impact → zero coefficient
Industry Code_Risk_Level	0.000000	No impact → zero coefficient
Accident Date_Year	- 0.000000	No impact → zero coefficient
Assembly Date_Year	- 0.000000	No impact → zero coefficient
C-2 Date_Year	0.000000	No impact → zero coefficient
C-3 Date_Year	0.000000	No impact → zero coefficient
Days_between_c2_and_c3	- 0.00103	Very small, negative impact.
Days_to_assembly_date	- 0.001924	Very small, negative impact.
Missing_Count	- 0.725847	Important, but with a negative impact
First Hearing Date_Year	- 0.736473	More important, strong negative relationship

Table 4 - Lasso coefficients interpretation table

Method Type	***Filter***			***Wrapper***		***Embedded***		***Final***
Feature	***Spearman***	***G-Test***	***Cramér's V***	***RFE LightGBM***	***RFE LR***	***Lasso***	***CatBoost***	
Age at Injury	?	-	-	Yes	No	Yes	-	Yes
Alternative Dispute Resolution	-	no	No	-	-	-	No	No
Attorney/Representative	-	yes	Yes	-	-	-	Yes	Yes
Average Weekly Wage	?	-	-	Yes	Yes	?	-	Yes
Birth Year	?	-	-	No	No	No	-	No
Carrier Type	-	No	No	-	-	-	No	No
County of Injury	-	No	No	-	-	-	No	No
COVID-19 Indicator	-	No	Yes	-	-	-	No	No
District Name	-	No	No	-	-	-	No	No
Gender	-	No	No	-	-	-	No	No
IME-4 Count	?	-	-	Yes	No	Yes	-	No
Industry Code	-	No	No	-	-	-	Yes	No
Medical Fee Region	-	No	No	-	-	-	No	No
WCIO Cause of Injury Code	-	Yes	Yes	-	-	-	Yes	Yes
WCIO Nature of Injury Code	-	Yes	Yes	-	-	-	No	Yes
WCIO Part Of Body Code	-	Yes	Yes	-	-	-	Yes	Yes
Agreement Reached	-	No	Yes	-	-	-	No	No
Number of Dependents	?	-	-	Yes	No	No	-	No
WCIO Cause of Injury Code_Group	-	Yes	No	-	-	-	Yes	Yes
WCIO Nature of Injury Code_Group	-	No	Yes	-	-	-	No	No
WCIO Part Of Body Code_Group	-	Yes	Yes	-	-	-	Yes	Yes
Industry Code_N_Entities	?	-	-	Yes	No	No	-	No
Industry Code_Risk_Level	?	-	-	No	No	No	-	No
Missing_Count	?	-	-	Yes	Yes	Yes	-	Yes
Under Age	-	No	No	-	-	-	No	No
Age_is_null	-	No	No	-	-	-	No	No
days_to_assembly_date	?	-	-	Yes	No	?	-	No
days_between_c2_and_c3	?	-	-	Yes	No	?	-	No
days_to_first_hearing	?	-	-	Yes	No	?	-	No
Accident Date_Year	?	-	-	No	No	No	-	No
Assembly Date_Year	?	-	-	No	Yes	No	-	No

C-2 Date_Year	?	-	-	No	Yes	No	-	No
C-3 Date_Year	?	-	-	No	No	No	-	No
First Hearing Date_Year	?	-	-	Yes	No	Yes	-	Yes
had_a_hearing	-	Yes	Yes	-	-	-	No	Yes
has_IME_4_Count	-	Yes	Yes	-	-	-	Yes	Yes
Average Weekly Wage_is_null	-	Yes	Yes	-	-	-	Yes	Yes

Table 5 - Summary of results from feature selection techniques

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
Logistic Regression	0.370123	0.288224	0.290527	0.752582	-0.000879
KNN	0.354090	0.311122	0.316666	0.725795	0.188652
SVC	0.298214	0.253667	0.230777	0.739570	0.000518
Decision Tree	0.346121	0.343752	0.344656	0.687448	0.335997
Gaussian Naive Bayes Classifier	0.339141	0.484360	0.263195	0.553067	-0.001405

Table 6 - Performance scores without imbalanced learning techniques

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
Random Forest	0.422857	0.350467	0.366316	0.746050	0.169172
LightGBM	0.420790	0.351465	0.347240	0.768056	0.023386
AdaBoost	0.309447	0.330566	0.161141	0.206663	0.001948

Table 7 - Ensemble performance scores

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
LightGBM	0.383455	0.533994	0.412904	0.714323	0.055187

Table 8 – Hybrid Re-sampling performance scores

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
Random Forest	0.368808	0.441854	0.392823	0.681800	0.146421
LightGBM	0.385456	0.535879	0.396766	0.716720	0.055010
AdaBoost	0.263136	0.462203	0.136594	0.147112	0.098060

Table 9 – Oversampling performance scores

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
Random Forest	0.315669	0.525089	0.299237	0.576913	0.127092
LightGBM	0.304859	0.511338	0.288135	0.521822	0.142209
AdaBoost	0.040535	0.319543	0.024037	0.029125	0.012597

Table 10 – Under sampling performance scores

Model	Precision	Recall	F1-Score	Accuracy	Overfitting
Random Forest	0.390798	0.380801	0.374206	0.718184	0.171424
LightGBM	0.420790	0.351465	0.347240	0.768056	0.023386
AdaBoost	0.280144	0.423435	0.098355	0.073530	0.001460

Table 11 - Model Weights performance scores

ANNEX 1 - FIGURES

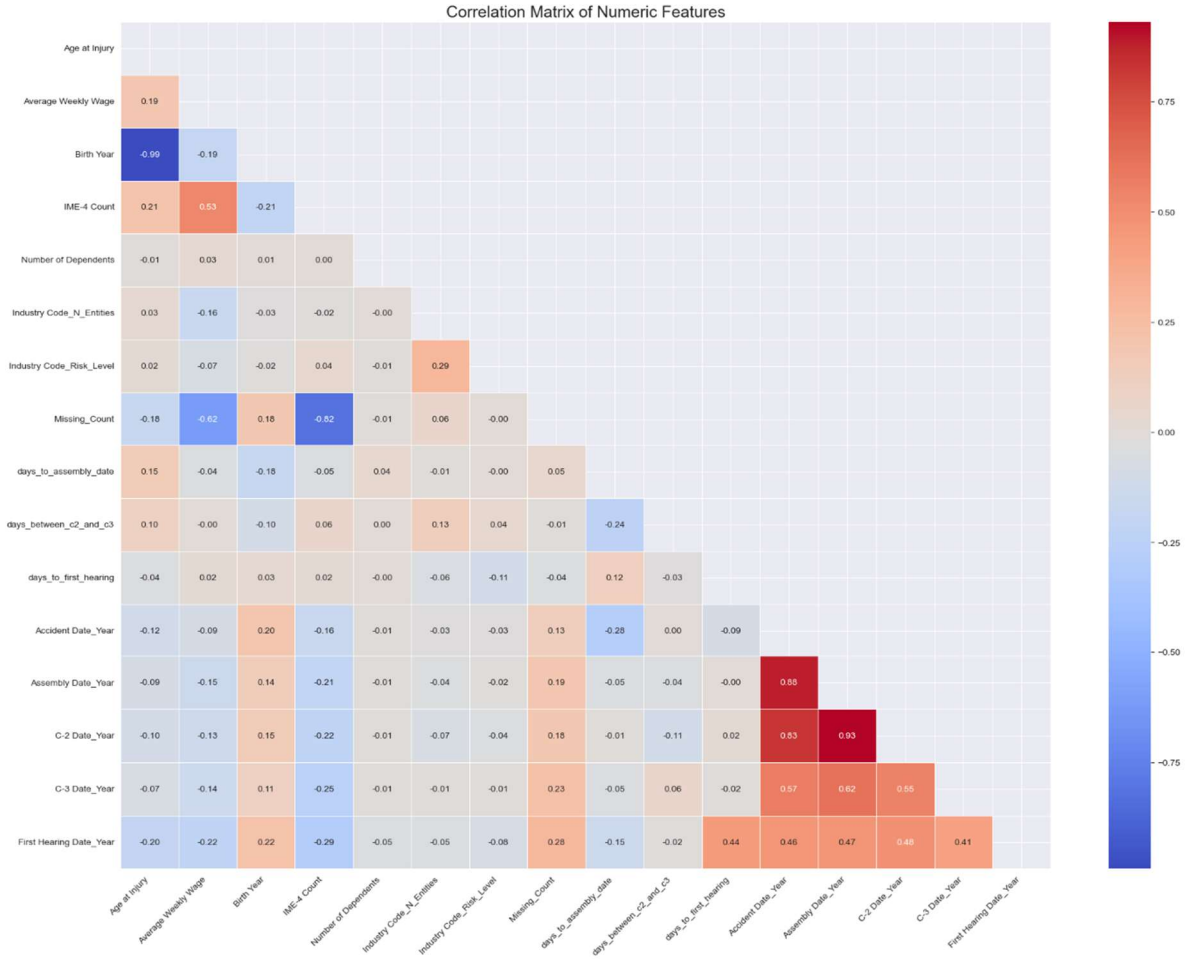

Figure 2 - Spearman Correlation Matrix

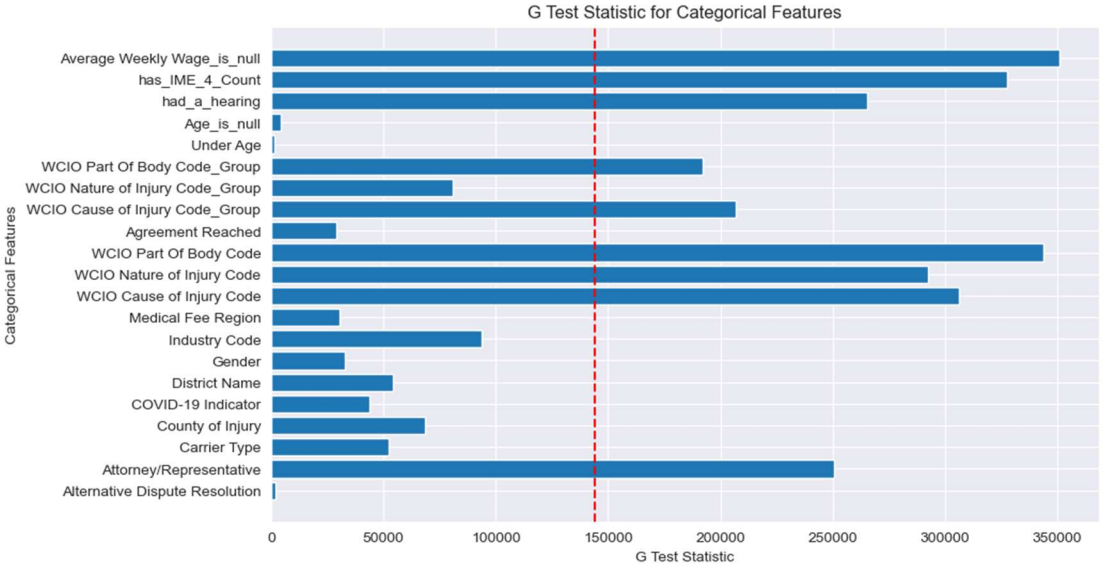

Figure 3 - G-test

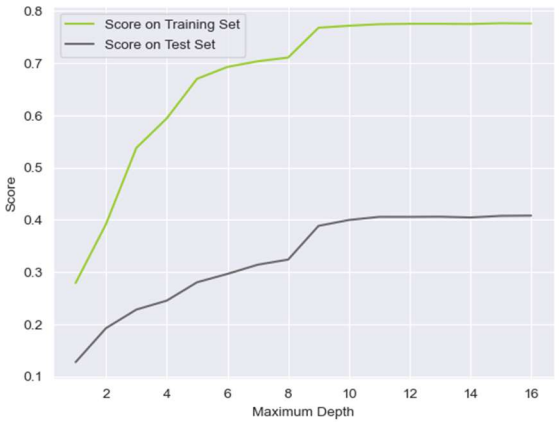

Figure 4 - Performance Curves for RFE with LightGBM

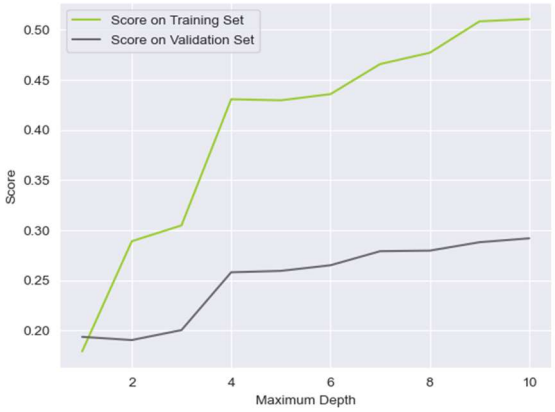

Figure 5 - Performance Curves for RFE with Linear Regression

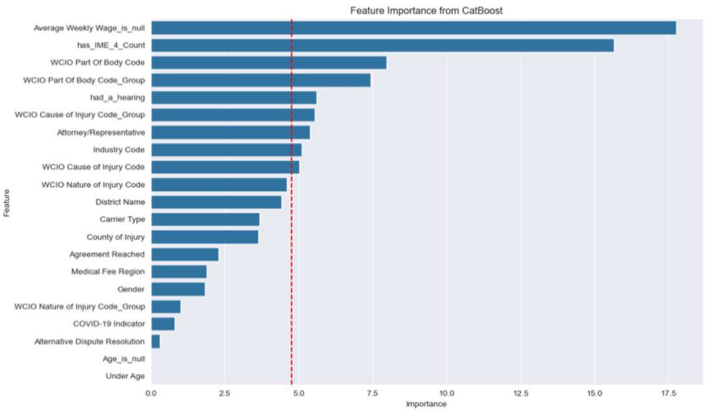

Figure 6 - CatBoost importance scores