MACHINE LEARNING PROJECT

Masters in Data Science and Advanced Analytics

NOVA Information Management School

Universidade Nova de Lisboa

Injury type prediction

Group 10

Joana Rodrigues, 20240603

Maria Francisca Fialho, 20240346

Rui Reis, 20240854

Tomás Silva, 20230982

Victor Pita, 20240596

Fall/Spring Semester 2024-2025

Table of contents

Introduction

The introduction for this project of machine learning consists of three main paragraphs. In the first one, the project and its goals will be explained briefly. The second paragraph describes the overview of the different parts made for the machine learning project. Finally, the last paragraph addresses the main challenge faced, how it was approached, what other researchers did when faced with the same problem and what we were expecting based on these findings.

This project's goal was to help WCB automate their process in terms of identifying the type of injury that occurred in the workplace, instead of having to do it manually. For that reason, a model was created to predict the type of injury (from light injury to death) caused based on certain factors, such as the date of the accident and the age of the individual.

The machine learning workflow is organized into five distinct notebooks:

1. **EDA and preliminary preprocessing**: exploratory data analysis of the different features of the dataset and initial data cleaning (including creation and removal of some features, filling of some missing values).
2. **Open ended section**: Prediction of the feature "Agreement Reached" through clustering. These predictions were later implemented in the test dataset for use in the 'Claim Injury Type' prediction.
3. **Standard split** (Figure 1): Training of different models with different hyperparameters combinations (including XGBoost, linear regression, MLP classifier and random forest) and using different feature selection techniques.
4. **StratifiedKFold** (Figure 2): concentrates on using the best hyperparameters and feature selection of notebook 3 to improve the XGBoost model with cross validation and by exploring different techniques to fight class imbalance. This ensured that the differences in the model's performance would come from the use of cross-validation rather than feature selection or hyperparameters of the models, maintaining consistency in the features and preprocessing strategies.
5. **Interface**: Interface created to predict directly the outcome of the model.

The main challenge found was to create a model capable of identifying all the seven different types of claims, since the dataset was very imbalanced. This was explored as well in "Solving the Problem of Class Imbalance in the Prediction of Hotel Cancellations: A Hybridized Machine Learning Approach" [1], a research paper where SMOTE-ENN is used to solve the previously mentioned problem. Based on their finding, our expectation was that using techniques of under sampling and/or oversampling such as SMOTE, class weighting, and outlier removal using DBSCAN and IQR would allow us to be able to predict all different types of injuries which was not entirely the case.

Data exploration and preprocessing

This section explains the steps taken regarding the exploration and preparation of the data (1st notebook). Several visualizations were created to understand the dataset and the variables that compose it. This analysis allowed us to gain an initial understanding of the dataset's structure and

identify potential preprocessing steps, required for optimal model performance. For the numerical features, the histograms used showed that some of the distributions of the features were right-skewed, indicating the presence of long tails with higher values. This pattern suggested the need for a log transformation, to normalize the data. For the categorical features, bar plots were done to visualize the frequency of each category. Furthermore, it was noticed that the data had several problems, and that they should be treated before using this data in the model, to improve its efficiency.

In addition to the box plots and bar plots for numerical and categorical features, respectively, it was decided to explore the relationship between the target variable and the features. This analysis led us to conclude that *Age at Injury* and some of the date features seem to be the most valuable for predicting the target among the numerical features. As for the categorical features including *Industry Code*, *District Name,* and other transformed features show the strongest correlations with the target.

In this section it will be explained how the issues observed were approached and solved, including the skewed distributions and missing values, the new features created and finally, the encoding and scaling techniques that were applied.

Firstly, **duplicates** were seen and removed through the claim identifier's repetition, as this should be a unique value, as it was set as the index of the dataset.

Then, the **datatypes** incorrectly attributed were replaced, since this change helps the model interpret data and process it accurately. Therefore, the following corrections were made:

- Dates were transformed from 'object' into 'date' features;
- Features with only two values (*Agreement Reached*; *COVID-19 Indicator*; *Attorney/ Representative*), were changed to Booleans.

Another preprocessing procedure done was to use existing **features** to **create** new ones, to try to improve the model's efficiency, as they may reveal previously hidden patterns:

Transformation	Columns created	New feature details
Groupings of the description's columns	*Industry Grouped*	Combination of industries into larger groups, depending on their nature.
	WCIO Cause of Injury Grouped	Similar causes of the injuries were grouped.
	WCIO Nature of Injury Grouped	Injury types were simplified into broader categories.
	WCIO Part of Body Grouped	Body parts were grouped according to their region on the body (example: 'Head and Neck').
Binary columns (indicating the missing values)	*C-2 Date Binary* *C-3 Date Binary* *First Hearing Date Binary* *Assembly Date Binary*	The absence of a date can mean that no action was taken, this might be a relevant insight. For example, this might happen in less severe injuries.
	IME-4 Count Binary nan	As most of the column is missing values, replacing them could introduce bias,

		therefore we record the absence information, as it may represent a pattern, and the column can't be used in other way.
Days columns (Days between the accident and the date in question)	*C-2 Days* *C-3 Days* *First Hearing Days* *Assembly Days*	To avoid having to work with dates. **Note**: If any of these dates were incorrectly entered (example: Accident happening after assembly date), they would have a negative value. To avoid this, a minimum value of 0 was set.
Accident date decomposition	*Day* *Month* *Year* *Day of the week*	To see if there is any pattern related to the accident date that can help predict the type of injury occurred, the date was decomposed into four different columns.
Log transformations	*Log Average Weekly Wage* *Log IME-4 Count* *Log Days* columns	To reduce the skewness in the data.

The existence of some errors and missing information led us to the **removal** of the following **rows**:

- Rows with less than 20% of the total information;
- Rows without '*Claim Injury Type'* (not possible to predict it if there is no value for it);
- Rows where individuals are under 14 years old (minimum legal age in the USA to work) and over 85 years old (likely errors or outliers).

In the cases where the **missing value** was assumed to be a specific value and not an imputation from the dataset, its filling was done before the data split.

- ***Age*** was filled with the *'Accident Date'* – *'Birth Year'*, for the cases where both these values existed, but '*Age at Injury Date'* didn't;
- The ***Log Days*** columns were filled with 0, since it was interpreted that the absence of dates indicates the event hasn't occurred;
- ***COVID-19 Indicator*** and ***Attorney/ Representative*** were considered to be 0 (*'No'*), since the absence of information can mean that it is not applicable in some cases. In any case, it is more neutral to assume that the individual doesn't have Covid or attorney than just *'Yes'*.

Having less dimensionality in the dataset helps save time, therefore, in some specific cases, some **features** were **removed** before the feature selection:

- Redundant features:
    - Description columns, since they have the same information as the *'Code'* columns.
    - *'Birth Year',* since it gives the same information as the '*Age'* column;
    - *'Date'* and *'Days'* columns, as this information was transformed into log columns, it would be redundant to keep both.
- Columns with only one unique value:
    - *'Assembly Date Binary'* is a column that evaluates whether a value was missing or not, and it originated from a column that didn't have any missing values, therefore, it's irrelevant;

    - *'WCB Decision'.*
- Columns that only exist in the training dataset:
    - *'WCB Decision',* to make sure that the model doesn't rely on unavailable information, during prediction.

Before splitting the data, the **target** variable ('*Claim Injury Type'*) was separated and encoded using a LabelEncoder. This was done since the target variable can be considered ordinal, as some injury types can be considered more severe than others. The data was initially split into 80% for training and 20% for validation, for further exploration (3rd notebook). In another approach (4th notebook), a stratified k-fold cross-validation with 3 folds was applied.

The remaining of the missing values were handled after the data split to prevent data leakage, as using information from the validation set gives the model information that should remain unseen when training. The following approaches were used:

- Numerical features were filled with the median of the training set to avoid the influence of outliers;
- Categorical features' missing values were substituted with the mode of the training set.

Finally, for the remaining features, the chosen encoder was CountEncoder, as it converts categorical values into numerical representations, based on their amount. Scaling was done with MinMaxScaler in all features, after the feature selection that will be explored in the next section. It was chosen because, even though there were some outliers within the data, the log-transformation applied to some features reduced this difference. Furthermore, this scaler works well with data that doesn't follow a normal distribution, which happens in some of the variables in the dataset.

Multiclass classification

Feature selection

When predicting an injury type is crucial to select the features that make the most sense, because reduces overfitting, especially on dataset big as this one, since "too many features lead to a model that is too complex and tailored to the training data, which means it is unlikely to generalize to new data" (Paulson et al., 2022, p. 6).

The feature selection methods used for categorical and numerical data tend to differ.

For categorical variables, Cramér's V was chosen and used to determine the correlation between two categorical variables based on a chi-squared test. The process was applied as follows:

1. Pairs of features highly correlated were identified (a correlation above 0,7 was considered to be highly correlated);
2. To ensure the least important features were the ones removed, the correlation of the correlated features with the target variable was compared;
3. The feature with the lowest correlation to the target variable was removed (Figure 3).

Regarding numerical and boolean features, the following methods were chosen:

Feature selection method	Reasoning for use
Correlation between features	Evaluates the redundancy between features.
Correlation with the target	Evaluates which features, from the ones highly correlated, should be removed.
Kendall p-value test	Resilient to outliers compared to other methods. It's helpful for features on our dataset that have high variance (including '*Age at Injury*', *'Average Weekly Wage'*, *'C2 Days'* and *'C3 Days'*).
RFE – logistic regression (with loop for number of features)	It helps evaluate the importance of each feature, based on its contribution to the model's performance.
Lasso-CV	Prevents overfit, giving less importance to highly correlated features, especially large datasets like ours, that have half a million records.
Ridge Regression	Performs well in datasets with highly correlated features.

A numeric/boolean feature was considered relevant and worth keeping if it was approved by, at least, one of the previous methods. This threshold was chosen after analyzing the results from each model and concluding that many features were being removed by all of them (Figure 4).

Model

Class Imbalance

Class imbalances are one of the biggest challenges in machine learning. In the current dataset, this challenge is present, as classes like '*7.PTD*' (encoded as 6), '*8.Death*' (encoded as 7), and '*6.PPD NSL*' (encoded as 5) make up less than 2% of the total data and, on the other hand, the category '*2.NON-COMP*' (encoded as 1) accounts for over 50% of the instances (Figure 5). With such an imbalance, the model naturally leans towards predicting the majority class, making it harder to identify the smaller ones. However, models not being able to predict certain classes, in a real-life scenario, may have serious consequences, such as misclassifying critical cases that require urgent attention or different resource allocation. To fight this, several approaches of under sampling of the majority classes and/or oversampling of minority classes were tried:

Type	Technique used	Reasoning	Disadvantage	Notebook
Under sampling by Removal of	**DBSCAN**	Clustering algorithm that can be used to detect and exclude noisy samples; Applied to remove outliers within each class to reduce class confusion (outliers from one class may resemble instances of another).	Too computationally expensive.	No

outliers in the majority class	**IQR**	Simplifies outlier removal using interquartile ranges to identify anomalies.	Results were worse due to over-removal.	No
	HDBSCAN	Hierarchical version of DBSCAN that is not as computationally expensive.	Not able to predict class 6 by itself (on most of the models).	Part 3 and 4
	Both (HDBSCAN + IQR)	Joint of two under sampling techniques to balance their strengths.	Too strict.	No
Random Under sampling	**Random Under sampling**	Simplifies, randomly removing from the majority class.	Risks losing important data.	Part 4
SMOTE	**SMOTE**	Generates samples for minority classes using Nearest Neighbors.	Overfits on minority classes.	Part 3 and 4
	SMOTE-ENN	Suggested in the research; Version of SMOTE that combines oversampling and cleaning.	Results weren't better than SMOTE and it's more computationally expensive.	Part 3
Class weights	**Class weights**	Adjusts the importance of each class during training, in some models, ensuring that minority classes receive more focus.	Limited effectiveness.	Part 3

Model Exploration and Selection

The process of model exploration and selection to predict the target variable '*Claim Injury Type*'. was divided into two phases:

1. Exploration of different models and their hyperparameters;
2. Tuning of the best model, applying cross validation and different ways to fight the class imbalance.

First Model Selection Round: Preliminary Exploration

Several models were tried to understand which would be able to better predict the data. The table below summarizes the performance of several different classifiers, tested without addressing the imbalanced classes:

Model Name	Accuracy_train	Accuracy_test	F1_score_train	F1_score_test	Hyperparameters
XGB	0.8135	0.7972	0.4863	0.4311	scale_pos_weight=class_weight_dict, n_estimators=500, learning_rate=0.1
KNN	0.840027	0.741481	0.5269	0.3776	n_neighbors=3
RF	0.8156	0.7946	0.3993	0.3705	n_estimators=100
BalancedRF	0.6204	0.6189	0.3217	0.318	n_estimators=100
MLP	0.7893	0.785462	0.3873	0.3775	hidden_layer=100,

					max_iter=300
LogisticR	0.7701	0.7706	0.3217	0.3212	max_iter=1000

All models were trained using the same preprocessing and features, and their performance was evaluated on both training and testing datasets. Among the classifiers, **XGB** stood out as the model with the highest test accuracy and F1 score.

From the exploration of the different models, several hyperparameters and techniques to address the class imbalance were tried. From this overview, the following conclusions were taken (further exploration of techniques for the class imbalance will be explained in the next section):

- It is evident that some models performed better under F1 Score when class weights were considered (Figure 6 - RF and RF_WC are both Random Forest models one with and the other without the weighted adjustment), although the difference was not substantial;
- Random Search is a less computational version of grid search that was applied to the models to improve the score based on choosing the best combination of hyperparameters;
- Using unusual techniques, such as outlier removal with DBSCAN for under sampling, may result in better performances than most common techniques (Figure 7).

Second Model Selection Round: Model tuning with cross validation

Based on the insights from the first round, the second round focused on further improving the XGBoost classifier using cross validation and different ways to fight the class imbalance, which was the biggest challenge of the dataset. The hyperparameters obtained through Random Search from the best-performing ones, from the first phase, were tried out with cross-validation. The following variations of XGBoost were tried:

- Simple XGBoost;
- XGBoost with random under sampling;
- XGBoost with random under sampling and SMOTE applied to class 6 (the smallest class);
- XGBoost with random under sampling, SMOTE for class 6, and HDBSCAN.

With these new techniques the results kept improving but at the same time overfitting became more evident, especially with SMOTE and HDBSCAN. These were tried mainly to predict class 6 but as the instances are very little, it overfitted a lot. The best balance of performance, considering overfitting, was with the model set featuring XGBoost and under-sampling (Figure 8). On the other hand, Cross-validation helped check the model in terms of coherence; yet some problems persisted regarding overfitting, even for the model chosen (Figure 9 – Performance).

Open ended section

After debating what made sense to explore for this section, two different approaches were decided:

1. **Clustering**: Using unsupervised learning in a supervised problem, by trying to predict the variable *'Agreement Reached'*.
2. **Interface**: Developing an interface, something focused on the users and the benefits they would receive from our project and the model we developed.

Clustering

Clustering is an unsupervised learning technique that groups similar instances. Even though our problem consists of supervised learning, clustering can be adapted by grouping the outcomes that are intended to be predicted into the same clusters and then determining which cluster the testing instances belong to. **This approach not only offers an atypical perspective but also highlights the application of interdisciplinary knowledge, incorporating concepts learned in other subjects into the context of this project.** Our goal was to cluster the outcomes of the feature 'Agreement Reached' into two distinct groups, each corresponding to instances of the same agreement outcome: "Yes" and "No". The following steps were taken:

1. **Preprocessing**: Only numeric columns are considered for this approach, the missing values and scaling are dealt the same way as previously mentioned;
2. **Feature selection**: The features were selected according to their relationship with the target (threshold of 20%);
3. **Feature Removal**: The *'Injury Claim Type'* feature was excluded because it was not available in the test dataset (*df_test*), and its inclusion could have introduced bias into the analysis;
4. **Cluster Creation**: Two clusters were created, each containing instances from a specific outcome of the target variable. In other words, one cluster only contained instances where the target variable was *'True'*, and the other only contained instances where it was *'False'*;
5. **Algorithm Selection**: The clustering algorithms chosen were K-Means and Gaussian Mixture Models (GMM). These algorithms calculate centroids that will represent each of the classes, to predict the different instances in the test dataset. For this reason, Hierarchical Clustering and DBSCAN weren't used, as it would be necessary to calculate pseudo-centroids to be able to predict them the same way;
6. **Cluster Assignment**: For the test data, each instance was compared to the centroids, and it was assigned to the cluster of the closest centroid.

Clustering	Mechanism	Advantages	Disadvantages	Results
K-means	Measures the distances between data points and assigns each point to the nearest centroid.	Easy to understand.	Measures the distances; Sensitive to outliers.	**Accuracy**: 0.7753 **F1 Macro (Test)**: 0.5557
GMM				**Accuracy**: 0.7290

	Models the data, using a mixture of Gaussian distributions, and assigns data points based on probability distributions.	More robust to outliers.	More computationally expensive; May struggle with the high dimensionality.	**F1 Macro (Test)**: 0.5364

As it's possible to see, the results obtained were not the best. The main reason behind this is that the imbalance between both outcomes goes unnoticed in K-Means and GMM, since they rely on centroids or statistical measures, and do not take into account the quantity of instances in each class. Since there is a much higher likelihood of the outcome to be *'False'* (95% of outcomes) rather than *'True'* (Figure 10) the clusters should account for this imbalance, to decrease misclassification errors. This is implemented by adjusting the predictions and applying a higher probability threshold for the occurrence of the '*False'* class (Cluster 0). After trying some different threshold values, the optimal threshold was determined to be the one that achieved a good balance between the accuracy rate and the F1 score. In most cases, increasing the threshold value (thus making it more likely to classify as the majority class) resulted in a decrease in the F1 Score (Macro) while increasing the accuracy (Figure 11). (The accuracy metric is particularly important in this context since it is directly used to predict the *'Claim Injury Type'*). Selecting an optimal threshold value was essential and was achieved by finding the best balance between accuracy and F1 score to ensure predictions for both classes while also guaranteeing that these predictions are accurate enough for use in the main model. Ultimately, the best-performing model was GMM with a 90% probability threshold applied to the *'False'* class, as it provided more reliable predictions overall.

Interface

Besides the clustering, it was also decided to explore an interface for the open-ended section. The developed interface's goal is to predict instances based on the trained XGBoost model (the best-performing model from notebook 3) through a Python library called **Streamlit** (package installed through the terminal). It utilizes the encoder, scaling, and feature selection techniques from the same notebook, all managed through various pickle files. When data is entered into the interface done, the appropriate pickle file is loaded, and the model's pre-configured parameters are applied to automatically predict the injury type. This process provides an interactive and dynamic prediction experience, offering real-time outputs for the likely injury type based on the submitted values.

Conclusion

In conclusion, this project's goal is to develop a machine learning model for automating the classification of workplace injuries for the WCB. Rather than attempting to implement every possible technique, a reverse approach was taken by first creating a baseline model and setting up a process to quickly develop a model with a good score, not only on train but also on validation.

Despite not achieving a perfect performance, especially in predicting minority classes (specifically Class 6, that couldn't be predicted at all), xgboost with random under sampling model holds significant potential to enhance the WCB's claim processing system especially when the injury type is less severe (as the classes that aren't being predicted well are the more severe cases).

The primary challenge encountered was balancing class imbalances while simultaneously preventing overfitting, a problem commonly faced in machine learning tasks dealing with highly imbalanced datasets, as highlighted by Li, Kamnitsas, and Glocker (2020), who analyzed overfitting under class imbalance in neural networks" [3]. Efforts to address class imbalance often led to overfitting, highlighting the importance of finding an optimal balance to ensure the model remains both robust and generalizable.

This project was essential for consolidating and giving a deeper understanding of machine learning techniques by putting into practice what was learned in class. Furthermore, by using clusters, we used interdisciplinary knowledge, incorporating concepts learned in other subjects into the context of this project. Finally, this groupwork was a valuable experience for us as aspiring data scientists, allowing us to tackle a meaningful real-world problem while refining our skills in predictive modeling and class imbalance management.

Annexes

[1] Adil, M., Ansari, M. F., Alahmadi, A., Wu, J. Z., & Chakrabortty, R. K. (2021). Solving the problem of class imbalance in the prediction of hotel cancelations: A hybridized machine learning approach. *Processes*, *9*(10), 1713.

[2] Paulson, N. H., Kubal, J., Ward, L., Saxena, S., Lu, W., & Babinec, S. J. (2022). Feature engineering for machine learning enabled early prediction of battery lifetime. *Journal of Power Sources*, *527*, 231127.

[3] Li, Z., Kamnitsas, K., & Glocker, B. (2020). Analyzing overfitting under class imbalance in neural networks for image segmentation. *IEEE transactions on medical imaging*, *40*(3), 1065-1077.

Figures

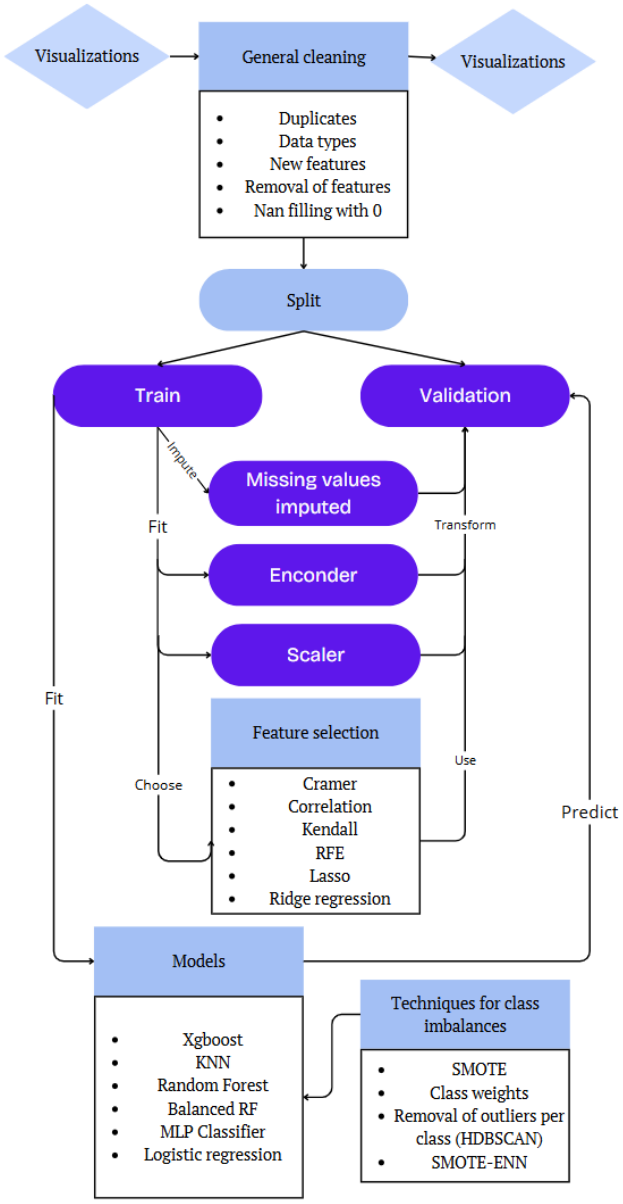

Figure 1: Overview of the model without cross validation.

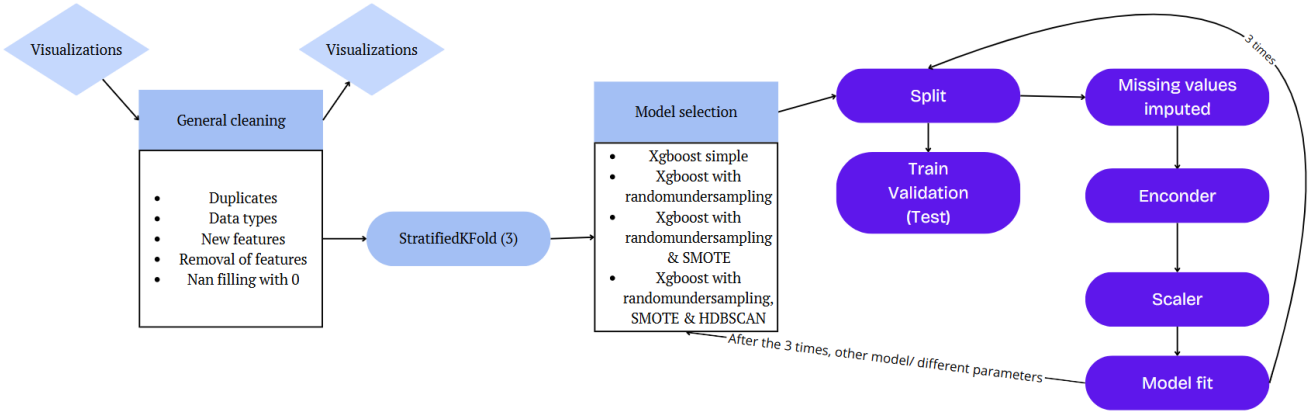

Figure 2: Overview of cross validation.

	Cramers V
WCIO Cause of Injury Code	True
County of Injury	True
WCIO Part Of Body Code	True
WCIO Part of Body Grouped	False
Carrier Name	False
Carrier Type	True
WCIO Nature of Injury Code	True
Industry Grouped	False
WCIO Cause of Injury Grouped	False
Zip Code	True
WCIO Nature of Injury Grouped	False
Industry Code	True

Figure 3: Categorical features chosen by Cramers V.

	RFE LR Evaluation	Lasso coefficient	Kendall	Ridge	Decision
Age at Injury	False	False	False	False	False
Alternative Dispute Resolution	False	False	False	False	False
Attorney/Representative	False	False	True	True	True
District Name	False	False	False	False	False
Gender	True	False	False	False	True
Medical Fee Region	False	False	False	False	False
Number of Dependents	False	False	False	False	False
log Average Weekly Wage	False	False	True	True	True
Log C-2 Days	False	False	False	False	False
Log C-3 Days	False	False	True	False	True
Log First Hearing Days	False	False	True	True	True
Log Assembly Days	False	False	False	False	False
Day of Week	False	False	False	False	False
Day of Month	False	False	False	False	False
Month	False	False	False	False	False
Year	False	False	False	False	False
Agreement Reached	False	False	True	True	True
C-2 Date Binary	False	False	False	True	True
C-3 Date Binary	False	False	True	True	True
First Hearing Date Binary	False	False	True	True	True

Figure 4: Feature selection of the numeric variables.

Claim Injury Type	Count	Percentage (%)
2. NON-COMP	287984	50.673395
4. TEMPORARY	147911	26.026281
3. MED ONLY	68640	12.077830
5. PPD SCH LOSS	48248	8.489673
1. CANCELLED	10761	1.893495
6. PPD NSL	4207	0.740260
8. DEATH	466	0.081997
7. PTD	97	0.017068

Figure 5: Class imbalances.

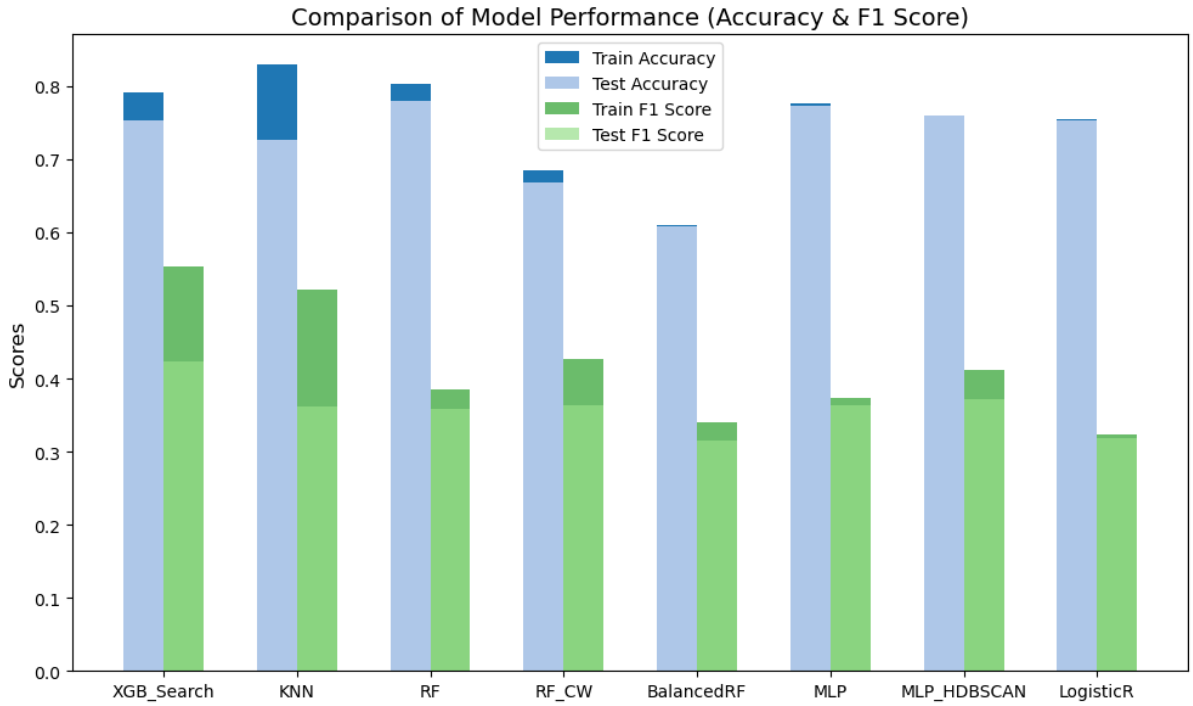

Figure 6: Difference between random forest with and without class weights.

Model Name	Accuracy_train	Accuracy_test	F1 score_train	F1_score_test	Hyperparameters
XGB	0.813345	0.798536	0.487672	0.422027	scale_pos_weight=class_weight_dict, n_estimator=500, learning_rate=0.1
KNN	0.840027	0.741481	0.530516	0.365277	n_neighbors=3
RF	0.802525	0.780465	0.385126	0.358601	n_estimator=100,
RF_CW	0.684954	0.668802	0.426042	0.363453	class_weight=class_weight_dict, n_estimator=100,
BalancedRF	0.609829	0.607981	0.339523	0.314955	n_estimator=100,
MLP	0.776077	0.773418	0.373575	0.362837	hidden_layer=100, max_iter=300
MLP_HDBSCAN	0.745866	0.759121	0.410966	0.371671	hidden_layer=100, max_iter=300
LogisticR	0.755362	0.753640	0.324161	0.317998	max_iter=1000

Figure 7: Example of different tries in one of the models.

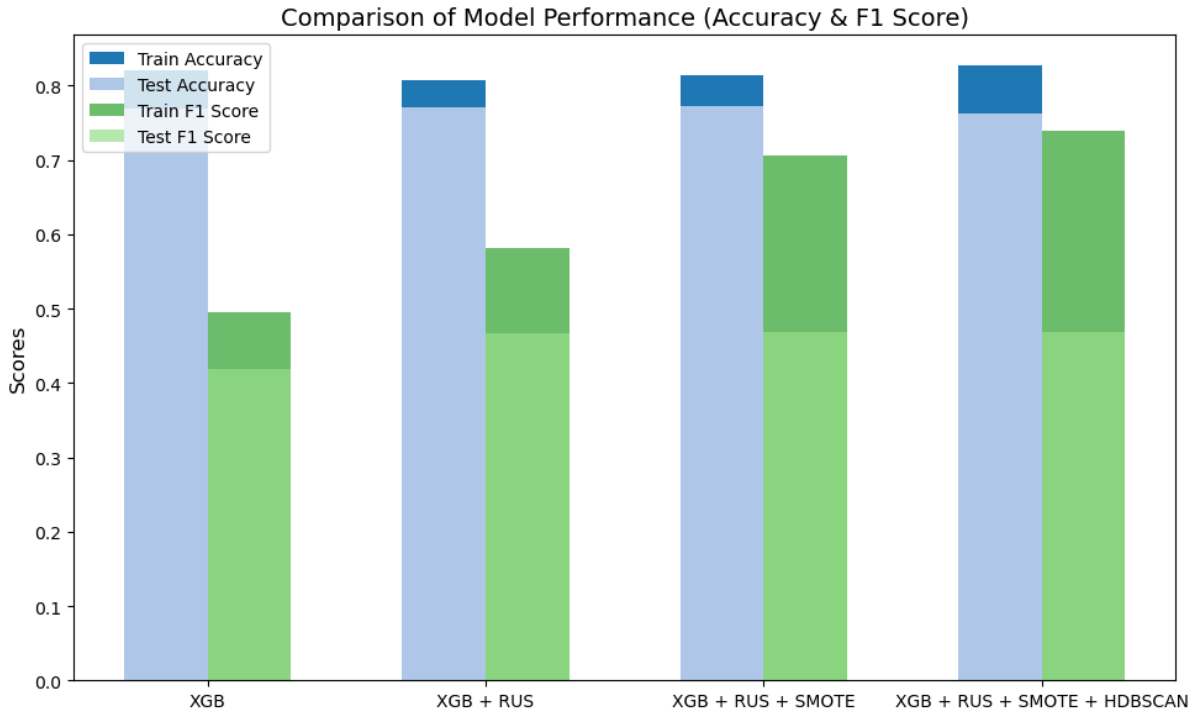

Figure 8: XGB with class imbalances strategies.

```
Accuracy of train: 0.8082
Accuracy of test: 0.7715
F1 Macro (Train): 0.5810
F1 Macro (Test): 0.4669

Classification Report for Validation Data:
              precision    recall  f1-score   support

           0       0.77      0.46      0.58      2666
           1       0.84      0.98      0.91     50000
           2       0.50      0.16      0.24     13334
           3       0.66      0.74      0.69     13333
           4       0.67      0.79      0.73     10000
           5       0.36      0.07      0.12      1402
           6       0.00      0.00      0.00        33
           7       0.61      0.39      0.47       155

    accuracy                           0.77     90923
   macro avg       0.55      0.45      0.47     90923
weighted avg       0.74      0.77      0.73     90923

\Confusion Matrix for Validation Data:
[[ 1235  1260   145    10    12     1     0     3]
 [  328 48968   595    65    33     0     0    11]
 [   29  7154  2105  2788  1213    37     0     8]
 [    4   373   881  9805  2153   102     0    15]
 [    6   257   399  1430  7876    31     0     1]
 [    1    26   102   760   415    98     0     0]
 [    0     0     5    22     4     2     0     0]
 [    1    18    14    58     1     3     0    60]]
```

Figure 9: Model with the best performance (still not able to predict Class 6).

Agreement Reached	Count	Percentage (%)
False	541909	95.353801
True	26405	4.646199

Figure 10: *'Agreement Reached'* imbalance.

Model	Test Accuracy	Test F1 Score
GMM_Model	0.729004	0.536370
GMM_Model_80	0.746988	0.542244
GMM_Model_85	0.865914	0.584231
GMM_Model_90	0.901181	0.582749
KMeans_Model	0.775299	0.555716
KMeans_Model_70	0.847637	0.590215
KMeans_Model_75	0.847887	0.590389

Figure 11: Clustering prediction of *'Agreement Reached'* performances.