Machine Learning Project

Master Degree Program in Data Science and Advanced Analytics

TO GRANT OR NOT TO GRANT: DECIDING ON COMPENSATION BENEFITS

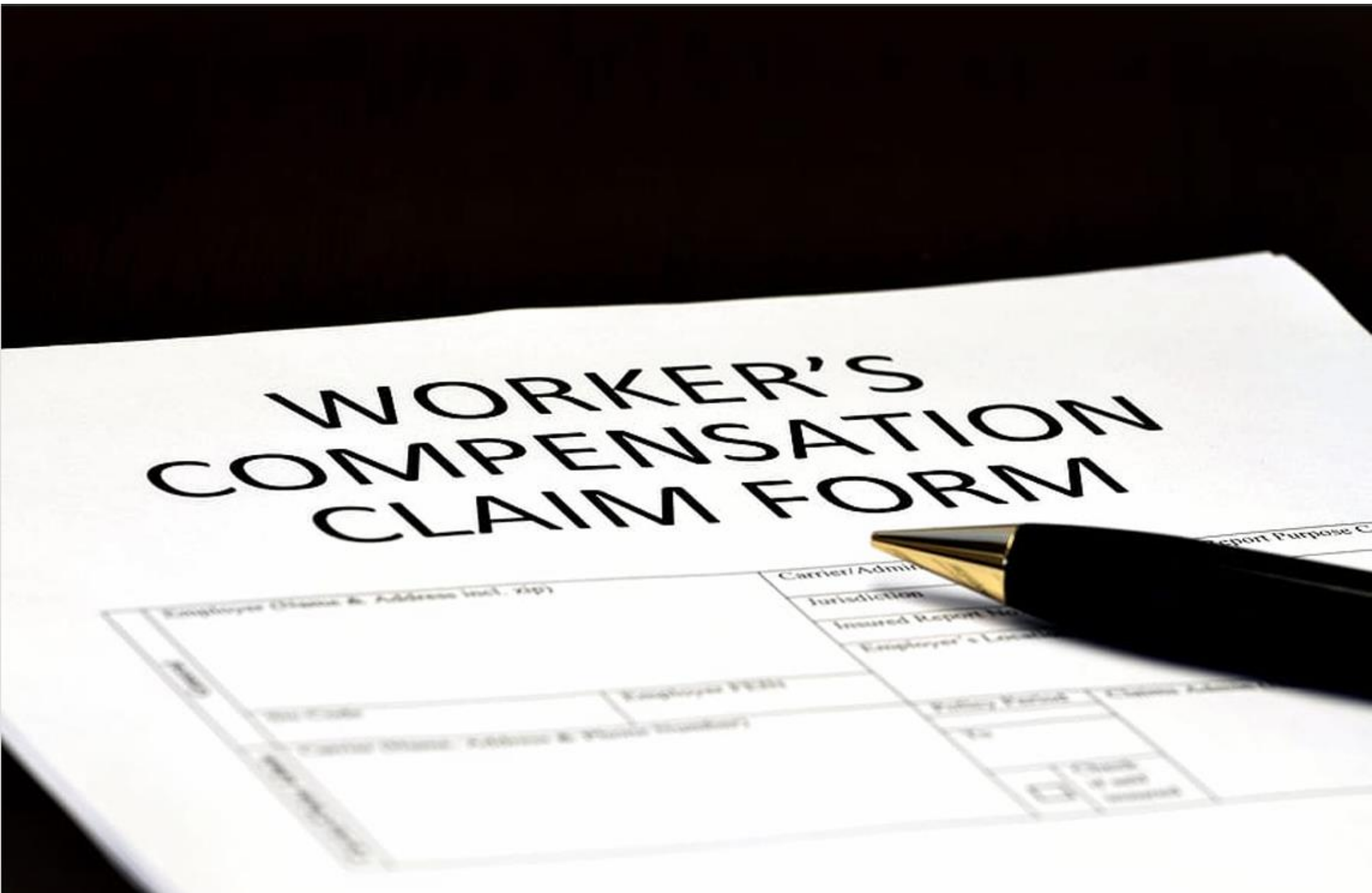
Group 22:

Beatriz Monteiro | 20240591

Catarina Nunes | 20230083

Jorge Cordeiro | 20240594

Leonor Wanzeller | 20240586



ABSTRACT

The New York Workers' Compensation Board (WCB) faces significant challenges in manually processing claims, a time-intensive task impacting efficiency. To address this, we developed machine learning models to automate the prediction of claim outcomes. Given these circumstances, we approached the claim injuries problem from two perspectives. Firstly, we developed a multiclass classification model capable of accurately predicting the WCB's final decision on claim injury types. Secondly, a binary model to predict if a worker would reach an agreement or not. Utilizing labeled data from claims filed between 2020 and 2022, this project includes an exploratory data analysis, rigorous data preprocessing, feature selection, and modeling. In the feature selection section, we implemented various techniques: Spearman correlations, LASSO, Mutual Information for encoded categorical columns, Decision Trees, and Random Forest. Finally, in the modeling section were tested several machine learning models such as Logistic Regression, K-Nearest Neighbors, Decision Trees, Gaussian Naive Bayes and MLP Classifier; and ensemble models such as Random Forest, AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost and Bagging Classifier. For multiclass, a voting classifier with Random Forest and LightGBM as base estimators produced the best results: the highest f1 score (0.4549). For the binary classification, a voting classifier with Logistic Regression, Gaussian Naive Bayes and Gradient Boosting as base estimators yielded the best results: the highest f1 score (0.6284).

For facilitating access, all project-related materials, including code and documentation, are available through the following resources:

Drive: Project Files

GitHub Repositories:

- Beatriz
- Catarina
- Jorge
- Leonor

Table of Contents

Table of Figures

Table of Contents

I. Introduction

The New York Workers' Compensation Board (WCB) regulates workplace injury claims to ensure workers receive benefits. However, reviewing over five million claims manually since 2000 is time-consuming and resource-intensive. To improve efficiency, the WCB is exploring automated solutions for classifying claims, particularly in predicting "Claim Injury Type" using historical data. Literature suggests that machine learning (ML) can effectively automate complex decision-making in this context:

In the study "*Fraud Detection and Analysis for Insurance Claim using Machine Learning*" concerning the insurance sector, the authors G. Hari Priya, K. Supraja applied machine learning models to classify claims into categories such as fraud detection, claim severity, or approval/denial predictions. Techniques like decision trees, random forests, and neural networks are frequently used for their ability to handle structured data.

The authors Samuel Tjahjono, Hendri Murfi and Sindy Devila study "Claim Severity Prediction of Workers' Compensation Using Tree and Neural Network Models" examines the application of models such as Decision Tree, Random Forest, XGBoost and Multi-Layer Perceptron (MLP) to predict workers' compensation insurance claims. The research compares the accuracies of these models, providing insights into their effectiveness in predicting claim severity.

Given the circumstances above, it is imperative for the WCB to have automatic solutions to predict claim injury outcomes. Thus, we aim to address such a problem through multiclass and binary classification models based on a dataset with claims filed between 2020 and 2022. First, a multiclass model that classifies our main target variable ***Claim Injury Type*** into seven categories: **'1. CANCELLED'**; **'2. NON-COMP'** (Non-Compensable); **'3. MED ONLY'** (Medical Only); **'4. TEMPORARY'** (Temporary Disability); **'5. PPD SCH LOSS'** (Permanent Partial Disability – Scheduled Loss); **'6. PPD NSL'** (Permanent Partial Disability – Non-Scheduled Loss); **'7. PTD'** (Permanent Total Disability); **'8. DEATH'.** Second, a binary model that predicts the ***Agreement Reached*** variable into two categories: **0** (no agreement was reached) and **1** (an agreement was reached).

Finally, given the results outlined above, we expect to deliver ML models capable of predicting claims fairly in both scenarios. Second, we aspire to propose models such as Logistic Regression, K-Nearest Neighbors, Decision Trees and MLP Classifier, but also ensemble models such as Random Forest, AdaBoost, Gradient Boosting, Bagging Classifier and Voting Classifier.

II. Data Exploration and Preprocessing

Data Characteristics and Initial Insights

The project's datasets consist of detailed information regarding claim injuries from the start of 2020 till the end of 2022 (training set) and from January 2023 onward (test set). This multivariate datasets include categorical columns (e.g., Carrier Type, Gender, etc.), codes columns (e.g., Industry Code, WCIO Cause of Injury Code, etc.), date columns (e.g., Accident Date, Assembly Date, etc.), discrete columns (e.g., Age at Injury, Birth Year, etc.), a continuous column (Average Weekly Wage) and Boolean columns (Agreement Reached, Attorney/Representative and COVID-19 Indicator). Check section A in the annexes for more details regarding the initial variables. Considering shapes, the training set has 593,471 entries and 33 variables and the test set has 387,975 entries and 30 variables.

Before applying any preprocessing to the variables, it is imperative to begin by exploring the characteristics of our data. We start by examining the data types, concluding that the numeric columns have inefficient datatypes; followed by a detailed univariate analysis that includes descriptive summaries of the features and an assessment of missing values, where we find that a lot of columns have the exact same number of missing values (19445) and it seems like they are just empty Claims; also, OIICS Nature of Injury Description, IME-4 Count, First Hearing Date, and C-3 Date have the majority of the rows of missing values (more than 50%).

We then move into data visualization, where numeric features are explored through histograms (*Figures 1 and 3*) and boxplots (*Figures 2 and 4*), categorical features are represented with bar plots (*Figure 5*) and geographic maps (*Figure* 6), and boolean features are analyzed with bar plots. Check section B in the annexes for more details regarding the EDA. Subsequently, we conduct a bivariate analysis to study correlations: examining relationships between numeric-numeric, numeric-categorical, and categorical-categorical features. Finally, a multivariate analysis addresses potential duplicate records.

Data Preprocessing

Regarding the missing values we study in the EDA, there are 19445 empty rows corresponding to empty claims, so we begin by removing these missing values and the column OIICS Nature of Injury Description, which has 100% missing values. Then we randomly split the dataset in a stratified way, which retained 70% of the data as a training set while the rest became our validation set.

In the Preprocessing section, we begin with an exploratory preprocessing notebook. This allows us to create a set of pipelines that can preprocess the datasets whenever needed, such as during the modeling functions. These pipelines include the following steps:

Table 1. *Data Preprocessing*

Pipeline	Section	Procedure
preprocessing_pipeline	*Binarization*	Change the columns 'Attorney/Representative' and 'COVID-19 Indicator' to binary indicators.
	Inconsistent Data	• When Gender = U or X, we change these values to NaN; also, when Alternative Dispute Resolution = U, we change to NaN; • If the Birth Year < 1934, we put Birth Year = 0; • When 'Age at Injury', 'Average Weekly Wage', 'Birth Year' were 0 we treat it as Nan. • Regarding the code's columns, we have the same description for multiple codes, so we standardize the code;
	Feature Engineering	• We are introducing a "**COVID Flag**" feature that helps the model identify patterns influenced by COVID-19. This flag indicates whether an accident occurred during the pandemic, separate from the COVID-19 Indicator column. • The **check_missing_dates** sees if any of the specified date columns are missing (NaN). If any are missing, it returns a comma-separated list of the missing columns; otherwise, it returns "OK". • **validate_dates** Applies three validation rules to the DataFrame and adds a flag for each one: (Rule 1) Checks if the "Assembly Date" is before the "Accident Date"; (Rule 2) Checks if either "C-2 Date" or "C-3 Date" is before the "Accident Date"; (Rule 3): Checks if either "C-2 Date" or "C-3 Date" is after the "Assembly Date". • **delay_days_category:** categorizes in ascending order the delay between Accident Date and Assembly Date into 5 groups; • **missing_info_category:** categorizes in ascending order the amount of missing values into 5 groups; • **Wage Category:** categorizes in ascending order the Average Weekly Wage into 6 groups; • **IME-4 Count Category:** categorizes the IME-4 Count into 6 groups in ascending order. Most of this categories was created to capture information that can influence the model even when the column has a lot of missing values (lile the wage and the IME-4 columns).
	Missing values	• For the categorical features we input the **most frequent value**; • For the numerical features we scale first and then input the **KNN**.

		It is important to note that the pipeline first processes the training data and saves the imputers and scalers to use them on the validation and test datasets, avoiding data leakage.
	Outliers	We handle outliers with the **Winsorization** method by setting lower and upper bounds for the training data using the Interquartile Range (IQR). Data points outside these bounds are adjusted to the new minimum and maximum values, and the same bounds are applied to the validation and test datasets.
	Mappings	We use predefined dictionaries to map the columns 'Industry Code Description', 'WCIO Nature of Injury Description', 'WCIO Cause of Injury Description', 'WCIO Part Of Body Description', and 'Carrier Type'. This approach is based on their distribution across the target variables, allowing us to **reduce dimensionality**.
	Variable pre-selection	We decided to drop the following columns: ['First Hearing Date', 'C-3 Date', 'IME-4 Count', 'Average Weekly Wage', 'Birth Year', 'COVID-19 Indicator', 'Accident Date', 'Assembly Date', 'C-2 Date', 'Carrier Name', 'County of Injury', 'District Name', 'Zip Code', 'Industry Code', 'WCIO Cause of Injury Code', 'WCIO Nature of Injury Code', 'WCIO Part Of Body Code', 'Industry Code Description', 'WCIO Cause of Injury Description', 'WCIO Nature of Injury Description', 'WCIO Part Of Body Description', 'Carrier Type']. We removed these columns because: (1) they **did not provide distinguishing information** that would help us differentiate the targets according to our criteria, and (2) we are using the **categorized or mapped versions** of these data points instead.
apply_groupings	*Grouping Categories*	In this step, we group categories of a feature **based on their distribution across the target variable**. This helps reduce the feature's complexity and dimensionality while retaining essential information for prediction.
encoder_and_scaler	*Encoding and Scaling*	• We use **ordinal encoding** for the columns: delay_days_category, IME-4 Count Category, Wage Category, and missing_info_category. • **One-hot encoding is applied to:** Medical Fee Region, Missing_Dates, Mapped Industry Code Description, Mapped WCIO Nature of Injury Description, Mapped WCIO Cause of Injury Description, Mapped WCIO Part of Body Description, and Mapped Carrier Type. • We scaled the numeric feature Age at Injury using the **Standard Scaler**, which is also set as the default option in our pipeline, allowing for the choice between it and the Min-Max Scaler.
reduce_memory_usage	*Data Types*	In this step, we optimize the DataFrame's memory usage, ensuring compatibility with algorithms like SMOTE. Continuous columns are treated as float32 or float64, while other columns are treated as int32 or int64.

It is important to note that, except the second pipeline (apply_groupings), all pipelines are identical for both multiclass classification and binary classification. In the second pipeline, we group the data based on the distribution of the target variable. In the multiclass scenario, we analyze the data across the eight Claim Injury Type categories. In contrast, for binary classification, we examine the two possible values (0 or 1) for the Agreement Reached column.

III. Multiclass Classification

The dataset presents a multiclass classification challenge due to significant class imbalance, particularly in the three least frequent target classes (5,6,7) that represent under 1% of the dataset. While techniques like SMOTE can help address this, their effectiveness may be compromised if feature selection overlooks columns with valuable information for these minority classes, simply because they contribute little to overall dataset variance.

Feature Selection

Defining the optimal subset of variables for the modelling stage is a crucial step, as this subset serves as the foundation for tuning and training the models. To achieve this, we employed **filter**, **wrapper**, and **embedded** methods to identify the optimal subset (Check Section E):

Table 2. *Feature Selection Approaches*

Method	Name	Actions
Filter methods	**Spearman Correlation**	Measures monotonic relationships between variables and helps exclude highly correlated variables (above a predefined threshold, in our case **0.8**). An exception is made for correlations with the target variable, as these are typically indicative of valuable relationships.
	Mutual Information	Evaluates the dependency between categorical data and the target variable. We set a threshold of **0.01 for general cases**. However, for the multiclass analysis, we **individually analyzed rare target classes** (5, 6, and 7). For these, we created an additional list that considers not only the general mutual information score but also variables relevant to the rare targets, applying a lower **threshold of 0.0001**.
Wrapper methods	**Decision Trees**	We performed a standard feature selection using '*class_weight = balanced*' for both multiclass and binary classifications, maintaining a **cumulative importance of 0.99**. For the multiclass case, we further analyzed the rare targets (5, 6, 7) by **treating them as binary targets**. We generated two lists of selected features: one based on the **weighted average of feature importance** (accounting for the target distribution) and another based on the **regular average**.
	Random Forests	A similar approach was applied, where we analyzed feature importance and created lists based on weighted and regular averages.
Embedded methods	**LASSO**	We excluded features with coefficients equal to zero, retaining only the important ones. Two lists were generated: one for **LASSO without SMOTE** and another for **LASSO with SMOTE**, for both multiclass and binary cases.

Modeling

The next step was to implement and develop multiclass models. We have created 3 pipelines:

Table 3. *Modeling Pipelines*

Modeling Function	Role
evaluate_model	This function performs cross-validation five times using **StratifiedKFold**. It separates the folds into training and validation sets and applies the preprocessing pipelines to each set (including SMOTE if selected). The function then fits the chosen model and provides performance metrics: **F1 macro averages** for both training and validation, as well as **accuracy**, **recall**, and **precision** solely for the validation set. Additionally, it calculates **overfitting** by subtracting the F1 score of the validation set from the F1 score of the training set and records the **time** taken to compute all these steps.
model_selection	This function calls the ***evaluate_model*** function and iterates over chosen **models**, **features**, scalers, and SMOTE options (if we want or not) based on the options we choose to evaluate.
optimize_hyperparameters	This function is used to find the **best parameters** for each model. It begins by splitting the dataset into features (X) and targets (y). Using *train_test_split*, the data is divided into a training set (X_train, 70% of the data) and a validation set (X_val). Next, the function **calls the preprocessing functions** and filters both the training and validation data according to the specified **feature set** in the parameters. An important step in this process is the use of the *PredefinedSplit* for the *cv* parameter in **GridSearchCV**. Since the dataset is already divided into training and validation sets, this approach ensures that we don't have data leakage during the cross-validation process. Finally, it executes **GridSearchCV** to perform a hyperparameter search to identify the optimal parameters for the model, **based on the macro F1 score of the validation set**.

After applying the first 2 functions for each combination of feature set and model, we ended up choosing the **Spearman feature set** for this analysis, as it consistently demonstrated good performance (compared to others) across the selected models.

Table 4. *Multiclass Classification Metrics for Decision-Making (Spearman Features, No SMOTE, Standard Scaler, and Outlier Treatment)*

Model	F1 score validation	F1 score train	Accuracy validation	Overfitting
Logistic Regression	0.382898965	0.385345889	0.764200165	0.002447
Decision Tree	0.36947619	0.975303732	0.676457166	0.605828
Random Forest	0.395922177	0.936128972	0.738602413	0.540207
Gradient Boosting	0.38549256	0.390712561	0.776965289	0.00522
AdaBoost	0.344065008	0.344094851	0.731207264	2.98E-05
XGBoost	0.429562	0.516624	0.781337	0.087062
LightGBM	0.387929775	0.428999502	0.769284874	0.04107
CatBoost	0.432177	0.518158	0.782104	0.085981

K Neighbors	0.395268866	0.479294885	0.74892	0.084026
Naive Bayes	0.075504937	0.075536195	0.110245634	3.13E-05
Neural Networks	0.421563195	0.491203614	0.780904577	0.06964
Bagging Classifier	0.398276	0.911492	0.727920	0.513216

For hyperparameter tuning, we selected the models highlighted in green. **Logistic Regression** was chosen strategically due to its moderate performance and low overfitting tendency. This characteristic makes it valuable in ensemble methods (such as voting and stacking), as it helps reduce the overall overfitting of the combined models.

The other models were selected based on their **macro F1 score** performance on the validation set, with one exception: **Gaussian Naïve Bayes**. This model was included despite its lower performance because we wanted to evaluate how much its contribution could improve the ensemble's performance. The **CatBoost** and **SVC** were excluded because they take too long to run.

Table 5. *Multiclass Classification with Hyperparameters for validation*

Model	F1 score validation	Accuracy validation	F1 score train	Overfitting	Time
Random Forest	0.439412	0.723143	0.668681	0.229269	1194.852
LightGBM	0.432287	0.784186	0.515715	0.083429	987.5
XGBoost	0.427999	0.782222	0.452419	0.024421	992.555
MLPClassifier	0.423735	0.780674	0.467701	0.043966	1508.334
Logistic Regression	0.398893	0.764478	0.402228	0.003336	2315.928

After hyperparameter tuning, cross-validation was applied to provide a more reliable F1 macro score. The Random Forest model performed well on smaller categories but showed signs of overfitting due to its tendency to memorize minority class patterns. Logistic Regression, with a high regularization value (C = 10), had a runtime of 15,000 seconds, so this parameter was removed for efficiency. We saw that the Min-Max Scaler tripled computation time, and SMOTE, which generates synthetic data for underrepresented classes, increased overfitting by amplifying noise – so in the final solution, we will use the Standard scaler and no SMOTE. XGBoost and LGBMClassifier showed similar results, with LGBMClassifier slightly better but more prone to overfitting. To address these issues, we chose Random Forest, LGBMClassifier, and Logistic Regression for ensemble strategies to balance performance and reduce overfitting.

Final Model Selection and Main Findings

In the final phase of the notebook, the focus shifted to combining the strengths of the best-performing models using ensemble techniques, specifically *Voting* and *Stacking* Classifiers.

We employed a *two-stage ensemble* approach in which the voting model first generated predictions, assigning specific classes to individual models: the Random Forest model handled classes *[0, 2, 4, 5, 6],* while the LGBM model managed *[1, 3, 7].* The final model chosen was the **Voting Classifier**, which combined Random Forest and LGBM with weights of (2, 1). This ensemble achieved a Macro Average F1-Score of 0.4549. Additionally, we analyzed the impact of incorporating the ***"Agreement Reached"*** variable, which exploratory data analysis (EDA) revealed to be correlated with class 3 ('4. TEMPORARY'). This incorporation improved the F1 score to 0.47. However, we encountered challenges related to class imbalance in binary predictions, which negatively affected results during Kaggle competition testing. Ultimately, we decided on the initial voting model using Spearman-selected features for its balanced performance.

IV. Binary Classification

Feature Selection

Similar to the multiclass scenario, the binary target value is significantly imbalanced; the target '1' accounts for less than 5% of the dataset. For more details, please refer to Section D in the annexes. We have detailed the process we followed in Table 2. Additionally, you can check Section F in the annexes to see which features each model retains and which ones it discards.

Modeling

For Binary classification, we used the same functions as in the Multiclass classification presented in Table 3.

Table 6. *Binary Classification Metrics for Decision-Making (Spearman Features, No SMOTE, Standard Scaler, and Outlier Treatment)*

Model	F1 score validation	F1 score train	Accuracy validation	Overfitting
K Neighbors	0.592161	0.942689	0.623673	0.031511
Random Forest	0.559429	0.951952	0.62654	0.067111
Bagging Classifier	0.558259	0.951356	0.625367	0.067108
Decision Tree	0.554154	0.951595	0.61765	0.063497
CatBoost	0.543326	0.955077	0.553523	0.010197
Gradient Boosting	0.543252	0.955129	0.543522	0.00027
XGBoost	0.541788	0.955242	0.546017	0.004229
MLP Classifier	0.53929	0.955289	0.539956	0.000666
LightGBM	0.539087	0.955302	0.540094	0.001008
Gaussian Naive Bayes	0.528877	0.715907	0.52894	0.000063
AdaBoost	0.50668	0.951112	0.506774	0.000095
Logistic Regression	0.501156	0.952138	0.500981	-0.000175

In this section, we also decided to use the Spearman feature set. We excluded CatBoost, MLP Classifier, and LightGBM due to their high computational time without a score that justifies the investment. The Bagging Classifier and Decision Tree were also excluded because they produced lower F1 scores and accuracy compared to K-Neighbors and Random Forest, and they exhibited more overfitting than K-Neighbors.

XGBoost was set aside in favor of Gradient Boosting, which offers similar performance (yet slightly better) while taking only half the time. We chose Gaussian Naïve Bayes and Logistic Regression due to their low overfitting tendencies, as they can be good options to combine with other models.

Table 7. *Binary Classification with Hyperparameters for validation*

Model	F1 score validation	Accuracy validation	F1 score train	Overfitting	Time
Gaussian Naive Bayes	0.610898	0.881843	0.610978	0.00008	758.439
K Neighbors	0.590285	0.9417	0.65049	0.06021	2131.12
Logistic Regression	0.56295	0.77261	0.563214	0.00026	661.35
Random Forest	0.559134	0.951777	0.627233	0.0681	627.404
Gradient Boosting	0.55679	0.952244	0.621533	0.06474	1641.6

After hyperparameter tuning, we tested the models again using cross-validation to determine the best combination of scaler and SMOTE. We chose the Standard Scaler over the Min-Max Scaler due to the latter's significantly higher computational time and opted not to use SMOTE to avoid the risk of overfitting. We eliminated K Neighbors because it was slower and performed worse than Gaussian Naïve Bayes in most metrics, except for accuracy, where it was still outperformed by Random Forest and Gradient Boosting.

Final Model Selection and Main Findings

Just like in the multiclass prediction, we tested some ensemble techniques. For the Voting Classifier, we used a combination of models: ['Logistic Regression', 'Gaussian Naive Bayes', 'Gradient Boosting'], with weights assigned as (2, 1, 2). This ensemble achieved a Macro Average F1-Score of 0.6284. The ensemble combined the strengths of the models as follows: **Gaussian Naive Bayes (GNB)**, the best-performing model, offered a good F1 score with minimal overfitting; **Gradient Boosting (GB)** had good accuracy, although it tended to exhibit more overfitting. To mitigate this, they were combined with **Logistic Regression** and GNB. The ensemble achieved an Area Under the Curve (AUC) of 0.87. However, challenges in predicting the minority class (class 1) persisted, as class imbalance remained a limiting factor despite the ensemble's improvements. Additionally, we tested the Stacking Classifier, but it was not as effective as the Voting Classifier in this case, so the final solution was the **Voting Classifier**.

V. Interface

The integration of Streamlit provides a user-friendly platform for collecting and processing input data, applying the preprocessing pipeline in real time to ensure accurate predictions. However, during implementation, the application crashed upon clicking the **Predict** button, likely due to resource limitations. Streamlit can struggle with large datasets or intensive computations on local machines. Using frameworks like Flask, Dash, or deploying on a cloud platform with scalable resources could address this issue. To verify that the problem is not in the code, we tested a simpler interface using Jupyter Notebook's widgets, which worked flawlessly, confirming that the issue is related to Streamlit's resource handling. You can check the streamlit here: https://claiminjurytypepredictionapp.streamlit.app

VI. Conclusion

As in every project, limited resources such as time and computational power often pose constraints. However, several avenues for future enhancements can be explored. One potential improvement is deploying the model on platforms like AWS SageMaker using machine learning operations tools such as Kubeflow, which can ensure scalability and efficient monitoring. Additionally, exploring advanced deep learning techniques with frameworks like PyTorch or Keras may enhance model performance, especially for complex datasets. Combining ensemble deep learning models could also provide more robust predictions. Furthermore, quantum machine learning, though still in its infancy, holds promise for revolutionizing data processing through faster computations and improved handling of complex datasets.

Despite our efforts, the results were not as promising as initially expected. The dataset presented significant class imbalance, which affected the model's ability to generalize effectively. Additionally, our models required considerable time to train, which limited the ability to experiment with a broader range of algorithms or hyperparameters. In a different setting, deploying the project on a cloud platform with scalable resources could have enabled faster experimentation and the opportunity to try more sophisticated models.

Section H of the annex complements this discussion by providing comparisons between models like XGBoost, CatBoost, and LightGBM, highlighting the differences between Gradient Boosting and XGBoost, and presenting illustrations of algorithms such as KNN, MLP, bagging, and decision trees.

References

1. G. Hari Priya, K. Supraja. (2023). *Fraud Detection and Analysis for Insurance Claims using Machine Learning. International Journal of Advanced Research in Science, Communication and Technology,* 14(9), 1–10. Retrieved from https://jespublication.com/uploads/2023-V14I904.pdf
2. S. Tjahjono, H. Murfi, and S. Devila, "Claim Severity Prediction of Workers' Compensation Using Tree and Neural Networks-Based Models," in *Proceedings of the 2024 11th IEEE Swiss Conference on Data Science (SDS)*, Zurich, Switzerland, May 2024. [Online]. Retrieved from https://ieeexplore.ieee.org/document/10675747
3. Ascenso, G., Del Ser, J., Kadow, C., Fister, D., Barriopedro, D., Restelli, M., Giuliani, M., & Castelletti, A. (2022). Analysis, characterization, prediction, and attribution of extreme atmospheric events with machine learning: A review. *arXiv*. Retrieved from https://arxiv.org/abs/2207.07580
4. Aggarwal, C. C. (2015). *Data mining: The textbook*. Springer. Retrieved from https://pzs.dstu.dp.ua/DataMining/bibl/Data%20Mining%20The%20Textbook.pdf

5. Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106. Retrieved from https://doi.org/10.1007/BF00116251
6. IBM. (n.d). K-Nearest Neighbors (KNN). Retrieved from https://www.ibm.com/topics/knn
7. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). LightGBM: A highly efficient gradient-boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 3146–3154. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf
8. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31, 6638–6648. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf
9. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. Retrieved from https://doi.org/10.1145/2939672.2939785
10. Li, C. (n.d.). A gentle introduction to gradient boosting. *College of Computer and Information Science, Northeastern University*. Retrieved from

https://www.khoury.northeastern.edu/home/vip/teach/MLcourse/4_boosting/slides/gradient_boosting.pdf

11. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics, 29*(5), 1189–1232. Retrieved from https://doi.org/10.1214/aos/1013203451
12. Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics, 28*(2), 337–374. Retrieved from https://doi.org/10.1214/aos/1016218223
13. Wang, Y., & Liu, J. (2024). A comprehensive review of quantum machine learning: From NISQ to fault tolerance. *arXiv*. Retrieved from https://arxiv.org/abs/2401.11351
14. Schuld, M., & Petruccione, F. (2023). Quantum machine learning: A review and case studies. *MDPI Entropy, 25*(2), 287. Retrieved from https://www.mdpi.com/1099-4300/25/2/287
15. Allcock, J., & Zhang, S. (2019). Quantum machine learning. *National Science Review, 6*(1), 26–39. Retrieved from https://academic.oup.com/nsr/article/6/1/26/5222655
16. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4171–4186. Retrieved from https://doi.org/10.18653/v1/N19-1423
17. Albuquerque, C., Henriques, R., & Castelli, M. (2022). A stacking-based artificial intelligence framework for the effective detection and localization of colon polyps. *Scientific Reports, 12*, Article 17834. Retrieved from https://doi.org/10.1038/s41598-022-21574-w

Annexes

Section A: Variables' Metadata

Table 8. *List of Variables*

Variable	Description	Type
Accident Date	Injury date of the claim.	*object*
Age at Injury	Age of injured worker when the injury occurred.	*float64*
Alternative Dispute Resolution	Adjudication processes external to the Board.	*object*
Assembly Date	The date the claim was first assembled.	*object*
Attorney/ Representative	Is the claim being represented by an Attorney?	*object*
Average Weekly Wage	The wage used to calculate workers' compensation, disability, or Paid Leave wage replacement benefits.	*float64*
Birth Year	The reported year of birth of the injured worker.	*float64*
C-2 Date	Date of receipt of the Employer's Report of Work-Related Injury/Illness or equivalent (formerly Form C-2).	*object*
C-3 Date	Date Form C-3 (Employee Claim Form) was received.	*object*
Carrier Name	Name of primary insurance provider responsible for providing workers' compensation coverage to the injured worker's employer.	*object*
Carrier Type	Type of primary insurance provider responsible for providing workers' compensation coverage.	*object*
Claim Identifier	Unique identifier for each claim, assigned by WCB.	*object*
County of Injury	Name of the New York County where the injury occurred.	*object*
COVID-19 Indicator	An indication that the claim may be associated with COVID-19.	*object*
District Name	Name of the WCB district office that oversees claims for that region or area of the state.	*object*
First Hearing Date	Date the first hearing was held on a claim at a WCB hearing location. A blank date means the claim has not yet had a hearing held.	*object*
Gender	The reported gender of the injured worker.	*object*
IME-4 Count	Number of IME-4 forms received per claim. The IME-4 form is the "Independent Examiner's Report of Independent Medical Examination" form.	*float64*
Industry Code	NAICS code and descriptions are available at: https://www.naics.com/search-naics-codes-by-industry/.	*float64*
Industry Code Description	A 2-digit NAICS industry code description is used to classify businesses according to their economic activity.	*object*
Medical Fee Region	Approximate region where the injured worker would receive medical service.	*object*
OIICS Nature of Injury Description	The OIICS nature of injury codes & descriptions are available at https://www.bls.gov/iif/oiics_manual_2007.pdf.	*float64*
WCIO Cause of Injury Code	The WCIO cause of injury codes & descriptions are at https://www.wcio.org/Active%20PNC/WCIO_Cause_Table.pdf	*float64*

WCIO Cause of Injury Description	The WCIO cause of injury descriptions are at https://www.wcio.org/Active%20PNC/WCIO_Cause_Table.pdf	*object*
WCIO Nature of Injury Code	The WCIO nature of injury codes are available at https://www.wcio.org/Active%20PNC/WCIO_Nature_Table.pdf	*float64*
WCIO Nature of Injury Description	The WCIO nature of injury descriptions are available at https://www.wcio.org/Active%20PNC/WCIO_Nature_Table.pdf	*object*
WCIO Part Of Body Code	The WCIO part of body codes are available at https://www.wcio.org/Active%20PNC/WCIO_Part_Table.pdf	*float64*
WCIO Part Of Body Description	The WCIO part of body descriptions are available at https://www.wcio.org/Active%20PNC/WCIO_Part_Table.pdf	*object*
Zip Code	The reported ZIP code of the injured worker's home address.	*object*
Agreement Reached	Binary variable: Yes, if there is an agreement without the involvement of the WCB -> unknown at the start of a claim.	*object*
WCB Decision	Multiclass variable: Decision of the WCB relative to the claim: "Accident" means that the claim refers to workplace accident, and "Occupational Disease" means illness from the workplace. -> requires WCB deliberation so it is unknown at the start of the claim.	*float64*
Claim Injury Type	**Main target variable**: Deliberation of the WCB relative to benefits awarded to the claim. Numbering indicates severity.	*object*
Number of Dependents	Number of dependents.	*float64*

Section B: Data Exploration

(information extracted from the notebook, histograms, boxplots, and bar plots below)

Table 9. *Findings & Inconsistencies*

Variable	Findings & Inconsistencies
Accident Date	3.9% missing values
Age at Injury	3.28% missing values; has *outliers* (the minimum is 0 and the maximum is 117); histogram shows a roughly normal distribution, with a peak around the ages of 25 to 59, slight right skew
Agreement Reached	3.28% missing values; binary; the bar plot suggests that the majority of claims do not result in an agreement (class imbalanced)
Alternative Dispute Resolution	3.28% missing values; 3 different values
Attorney/Representative	3.28% missing values; binary (Y or N); the bar plot suggests that the majority of claims do not involve an attorney
Average Weekly Wage	8.1% missing values; has outliers (wage of 0 to 2,828,079) which turns this distribution highly skewed to the right (*Figures 1 and 2*)
Birth Year	8.18% missing values; has inconsistent values (0 and 2018, considering the data is from claims between the start of 2020 and the end of 2022, it's not possible to have a 2-4-year-old
C-2 Date	5.73% missing values
C-3 Date	68.45% missing values

Carrier Name	3.28% missing values; contains a lot of different values (potential grouping or categorization)
Carrier Type	3.28% missing values
Claim Injury Type	3.28% missing values
County of Injury	3.28% missing values
COVID-19 Indicator	3.28% missing values; binary (Y or N); the bar plot suggests that a significant portion of the claims are not related to COVID-19
District Name	3.28% missing values
First Hearing Date	74.59% missing values
Gender	3.28% missing values; 4 different values
IME-4 Count	77.62% missing values; has outliers; The histogram is right-skewed, with most counts concentrated around the lower values (0 to 4).
Industry Code	4.95% missing values
Industry Code Description	4.95% missing values
Medical Fee Region	3.28% missing values
Number of Dependents	3.28% missing values; The histogram shows a flat distribution across categories 0 to 6, indicating uniformity in the dataset concerning dependents.
OIICS Nature of Injury Description	100.0% missing values
WCB Decision	3.28% missing values; Only one unique value
WCIO Cause of Injury Code	5.91% missing values
WCIO Cause of Injury Description	5.91% missing values
WCIO Nature of Injury Code	5.91% missing values
WCIO Nature of Injury Description	5.91% missing values
WCIO Part Of Body Code	6.15% missing values; has a negative value of -9 (codes are positive)
WCIO Part Of Body Description	6.15% missing values
Zip Code	8.1% missing values

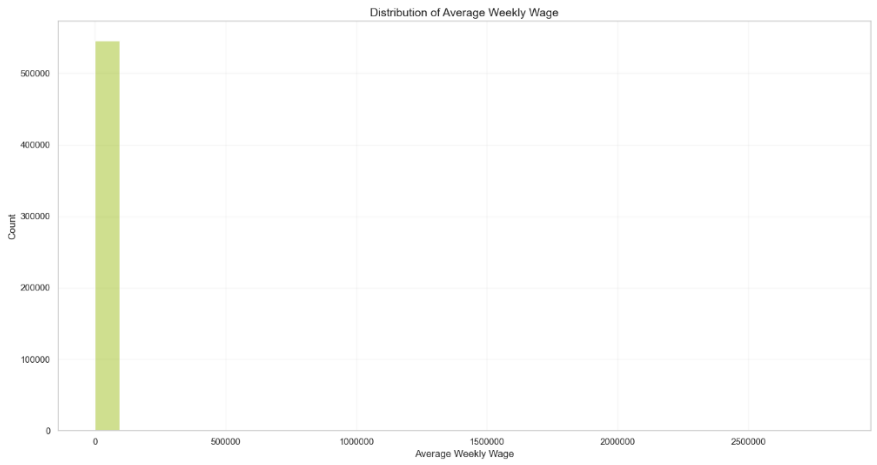


Figure 1. *Average Weekly Wage Histogram*

Table 10. *Average Weekly IQR values*

Variable	Min	Lower Fence	Q1	Median	Q3	Upper Fence	Max
Average Weekly Wage	0.0	-1261.5	0.0	0.0	841.0	2102.5	2828079.0

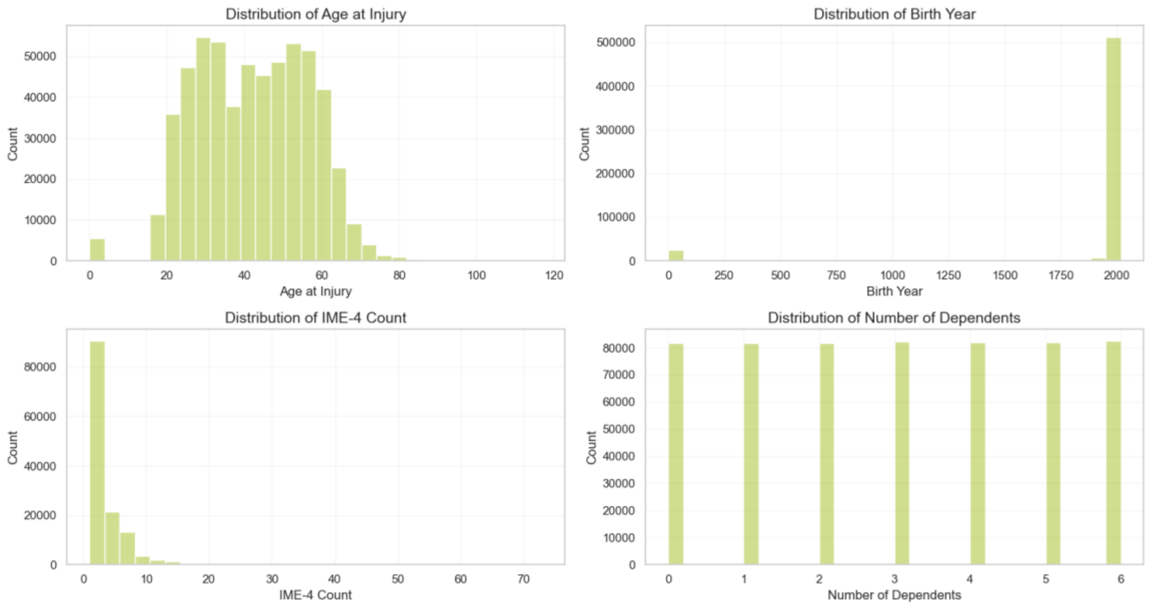


Figure 2. *Histogram for discrete columns*

Table 11. *IQR values for discrete columns*

Variable	Min	Lower Fence	Q1	Median	Q3	Upper Fence	Max
Age at Injury	0.0	-3.5	31.0	42.0	54.0	88.5	117.0
Birth Year	0.0	1929.0	1965.0	1977.0	1989.0	2025.0	2018.0
IME-4 Count	1.0	-3.5	1.0	2.0	4.0	8.5	73.0
Number of Dependents	0.0	-5.0	1.0	3.0	5.0	11.0	6.0

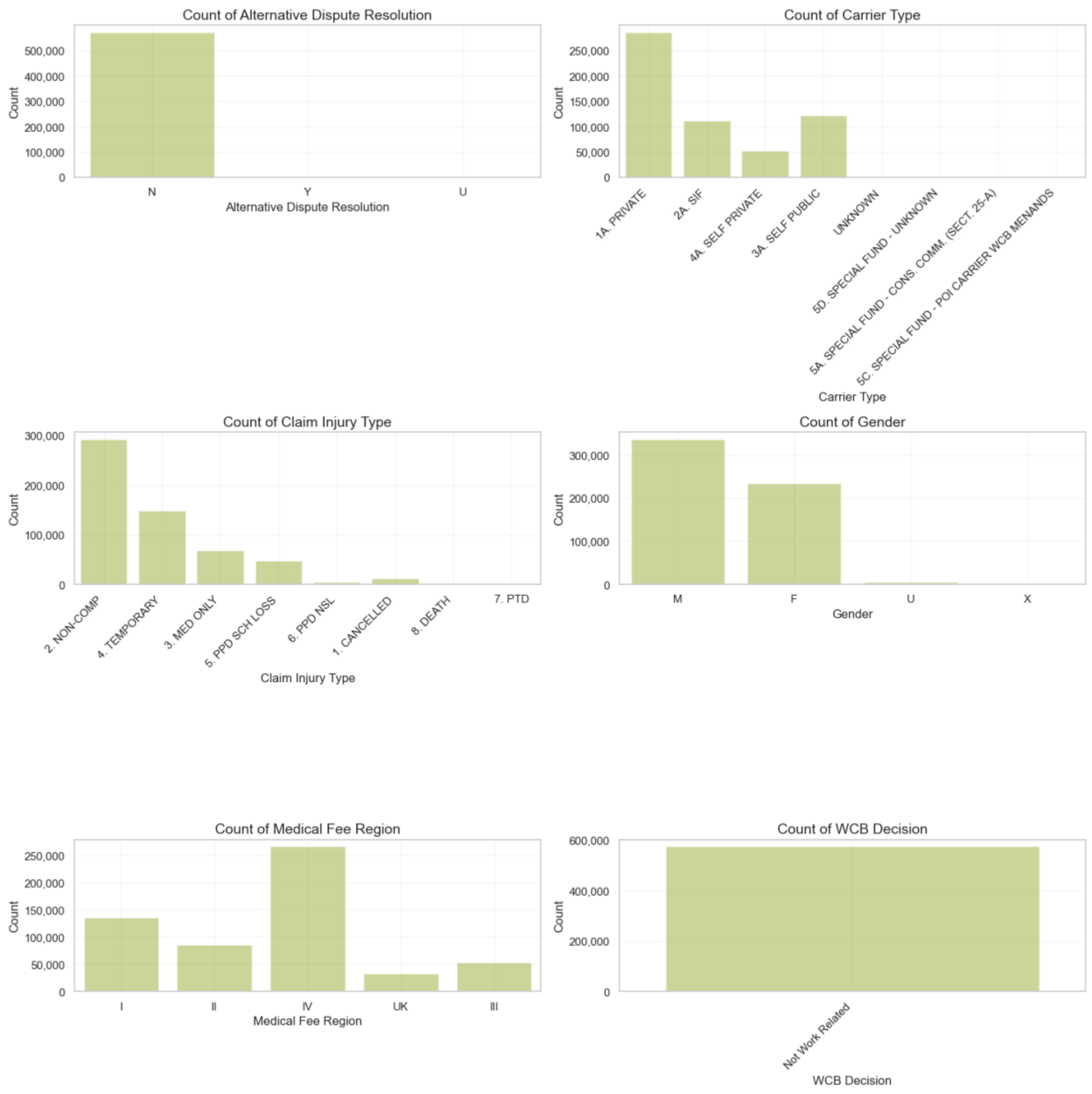


Figure 3. *Bar Plots for categorical variables*

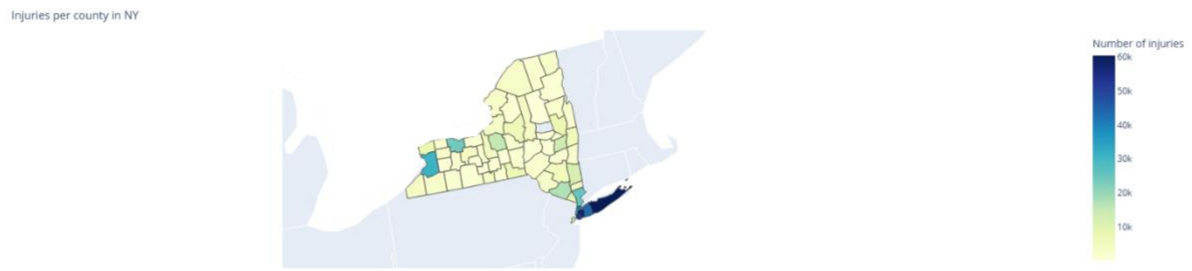


Figure 4. *Injuries per county in New York*

It's interesting to see that with the map we can see that the concentration of injury claims in Suffolk, Queens, Kings, Nassau, and Bronx can be attributed to a combination of geographical location, the presence of heavy industries, economic factors, and demographic characteristics. The proximity to ports and industrial zones often correlates with higher injury rates due to the nature of the work involved.

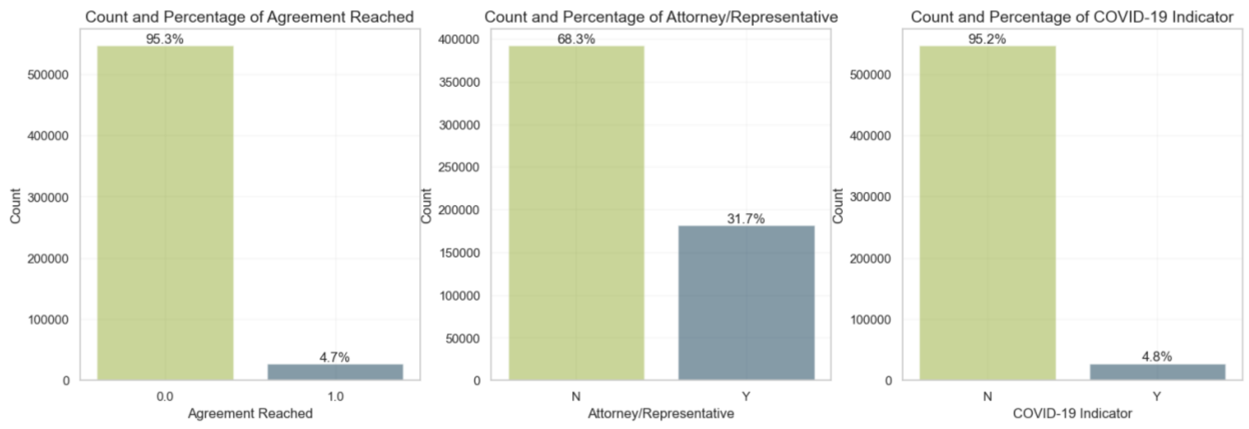


Figure 5. *Bar Plots for boolean variables*

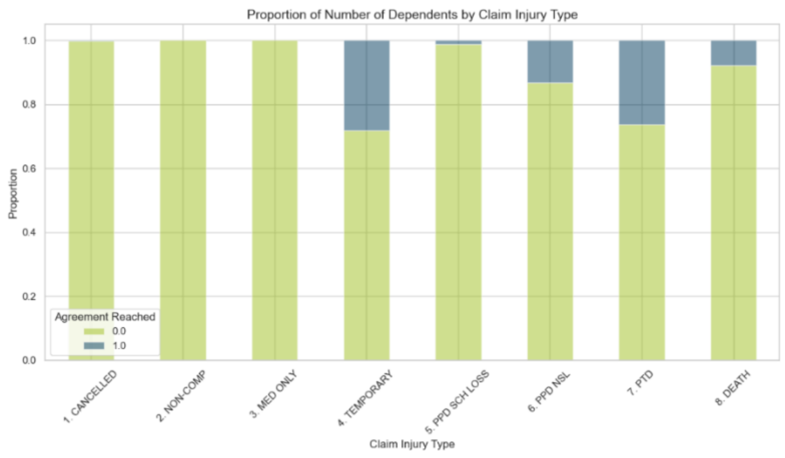


Figure 6. *Agreement Reached distribution across Claim Injury Type*

Section C: Target for multiclass classification

Table 12. *Frequency of each target value*

Claim Injury Type	Frequency
2. NON-COMP (1)	0.507080
4. TEMPORARY (3)	0.258712
3. MED ONLY (2)	0.120039
5. PPD SCH LOSS (4)	0.084108
1. CANCELLED (0)	0.021736
6. PPD NSL (5)	0.007337
8. DEATH (7)	0.000819
7. PTD (6)	0.000169

Section D: Target for binary classification

Table 13. *Frequency of each target value*

Agreement Reached	Frequency
0	0.953335
1	0.046665

Section E: Feature Selection & Modelling for Multiclass Classification

1. ***Filter methods*** evaluate the intrinsic properties of the data, such as correlations or dependencies between variables, to select features independently of any specific model.
2. ***Wrapper methods*** involve training a predictive model and selecting features based on their contribution to the model's performance.
3. ***Embedded methods*** combine feature selection with the model training process, using algorithms that inherently identify the most relevant features during optimization.

Table 14. *Feature Selection for Multiclass Classification*

Feature	Spearman	MI all	MI with rare	LASSO no SMOTE	LASSO with SMOTE	Standard DT	Weight DT	Mean DT	Standard RF	Weight RF	Mean RF
delay_days_category	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
IME-4 Count Category	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Wage Category	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
missing_info_category	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Medical Fee Region_Medical Group 1	Keep	Discard	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Medical Fee Region_Medical Group 2	Discard	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Medical Fee Region_UK	Keep	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Missing_Dates_Accident Date, C-3 Date	Keep	Discard	Discard	Keep	Keep	Discard	Discard	Discard	Discard	Discard	Discard
Missing_Dates_Accident Date, C-3 Date, First Hearing Date	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Missing_Dates_Accident Date, First Hearing Date	Keep	Discard	Discard	Keep	Keep	Discard	Discard	Discard	Discard	Discard	Discard
Missing_Dates_C-2 Related	Keep	Keep	Keep	Keep	Keep	Discard	Discard	Discard	Keep	Discard	Keep
Missing_Dates_C-3 Date	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Missing_Dates_C-3 Date, First Hearing Date	Keep	Keep	Keep	Keep	Keep	Keep	Discard	Keep	Keep	Keep	Keep

Missing_Dates_First Hearing Date	Keep	Keep	Keep	Keep	Keep	Discard	Discard	Keep	Keep	Keep	Keep
Missing_Dates_OK	Keep	Keep	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped Industry Code Description_Industry_high_0	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped Industry Code Description_Industry_high_1_2	Keep	Keep	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped Industry Code Description_Industry_high_3_2	Keep	Discard	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped Industry Code Description_Industry_high_4_5	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 0	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 1	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 10	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 11	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 12	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 13	Keep	Discard	Discard	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 14	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 15	Keep	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 16	Keep	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 17	Keep	Discard	Keep	Discard	Keep	Keep	Discard	Keep	Keep	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 2	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 3	Keep	Discard	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 4	Keep	Discard	Discard	Discard	Keep	Discard	Discard	Discard	Discard	Keep	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 5	Keep	Keep	Keep	Keep	Keep	Keep	Discard	Keep	Keep	Keep	Keep
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 6	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 7	Discard	Discard	Discard	Keep	Keep	Discard	Discard	Keep	Keep	Discard	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 8	Keep	Discard	Discard	Discard	Discard	Discard	Keep	Discard	Discard	Keep	Discard
Mapped WCIO Nature of Injury Description_Nature of Injury Cluster 9	Keep	Discard	Discard	Discard	Discard	Discard	Keep	Discard	Discard	Discard	Discard
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 0_14	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 1	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 10	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 11	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 12_6	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 13	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 15	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 2	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 3	Keep	Keep	Keep	Keep	Keep	Keep	Discard	Keep	Keep	Discard	Keep
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 4	Keep	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 5	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 7_8	Keep	Discard	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Cause of Injury Description_Cause of Injury Cluster 9	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Part Of Body Description_Part Of Body Cluster 0_12	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 1	Keep	Keep	Keep	Keep	Keep	Keep	Discard	Keep	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 10	Keep	Discard	Keep	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 11	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Part Of Body Description_Part Of Body Cluster 2	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 3	Keep	Discard	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 4	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 5	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Part Of Body Description_Part Of Body Cluster 6	Keep	Discard	Discard	Discard	Discard	Keep	Discard	Discard	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 7	Keep	Discard	Discard	Discard	Keep	Keep	Discard	Keep	Keep	Keep	Keep
Mapped WCIO Part Of Body Description_Part Of Body Cluster 8	Keep	Discard	Discard	Discard	Keep	Keep	Discard	Keep	Keep	Discard	Discard
Mapped WCIO Part Of Body Description_Part Of Body Cluster 9	Keep	Discard	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped Carrier Type_Private Carriers	Discard	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped Carrier Type_Public Carriers	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Mapped Carrier Type_SF_0_2	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped Carrier Type_SF_0_3	Keep	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard	Discard
Mapped Carrier Type_SF_2_1	Keep	Discard	Discard	Discard	Discard	Keep	Keep	Keep	Discard	Keep	Keep

Mapped Carrier Type_Unknown	Keep	Discard	Discard	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Keep
Age at Injury	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Alternative Dispute Resolution	Keep	Discard	Discard	Keep	Keep	Keep	Discard	Keep	Keep	Discard	Keep
Attorney/Representative	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Gender	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
C-2 or C-3 Date after Assembly Date	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
COVID Period	Keep	Discard	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep	Keep
Assembly Date or C-2 or C-3 Date before Accident Date	Keep	Discard	Discard	Keep	Discard	Discard	Keep	Keep	Keep	Keep	Keep

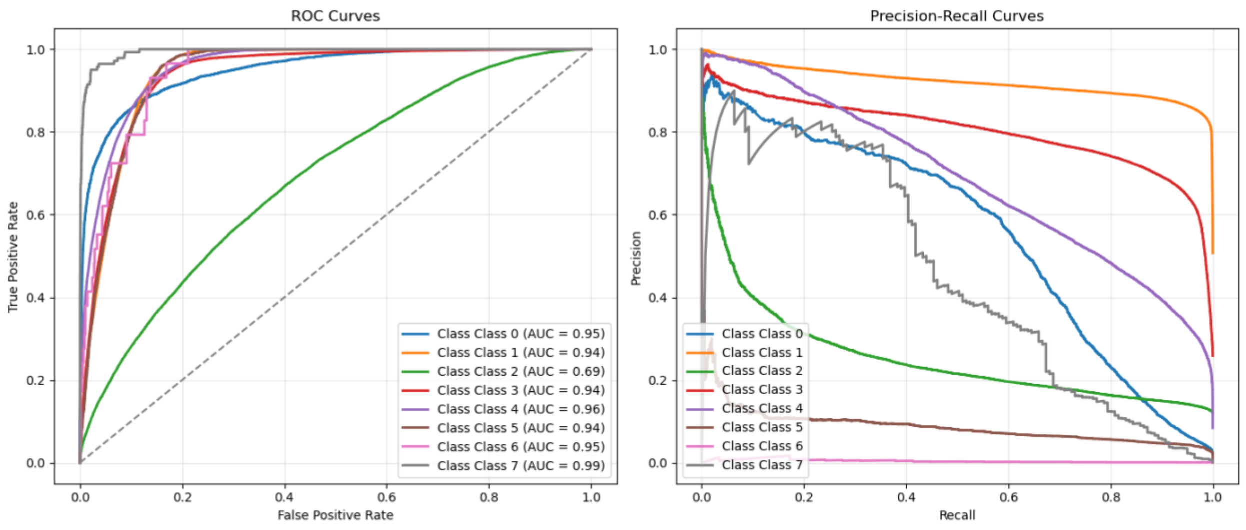


Figure 7. *ROC Curve and AUC & Precision-Recall Curve for each Class (Multiclass Model: Voting Classifiers)*

Section F: Feature Selection & Modelling for Binary Classification

Table 15. *Feature Selection for Binary Classification*

Feature	Spearman	MI all	LASSO no SMOTE	LASSO with SMOTE	Standard DT	Standard RF
Mapped Industry Code Description	Keep	Keep	Keep	Keep	Keep	Keep
Mapped WCIO Cause of Injury Description	Keep	Keep	Keep	Keep	Keep	Discard
Mapped WCIO Part Of Body Description	Keep	Keep	Keep	Keep	Keep	Keep
Medical Fee Region	Keep	Keep	Keep	Keep	Keep	Keep
delay_days_category	Keep	Keep	Keep	Keep	Keep	Keep
IME-4 Count Category	Keep	Keep	Keep	Keep	Keep	Keep
Wage Category	Keep	Keep	Keep	Keep	Keep	Keep
missing_info_category	Keep	Keep	Keep	Keep	Keep	Keep
Missing_Dates_missing_dates_0	Discard	Keep	Discard	Discard	Keep	Keep
Missing_Dates_missing_dates_1	Keep	Keep	Keep	Keep	Keep	Keep
Missing_Dates_missing_dates_high_0	Keep	Keep	Keep	Keep	Discard	Discard
Mapped WCIO Nature of Injury Description_nature_0	Discard	Keep	Keep	Keep	Discard	Discard

Mapped WCIO Nature of Injury Description_nature_1	Keep	Keep	Keep	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_nature_high_0	Keep	Discard	Discard	Discard	Discard	Discard
Mapped WCIO Nature of Injury Description_nature_high_1	Keep	Discard	Discard	Discard	Discard	Discard
Mapped Carrier Type_Carrier_0	Keep	Discard	Discard	Discard	Discard	Discard
Mapped Carrier Type_Carrier_1	Keep	Discard	Discard	Discard	Discard	Discard
Mapped Carrier Type_Carrier_high_0	Keep	Discard	Discard	Discard	Discard	Discard
Mapped Carrier Type_Carrier_high_1	Keep	Discard	Discard	Discard	Discard	Discard
Age at Injury	Keep	Keep	Keep	Keep	Keep	Keep
Attorney/Representative	Keep	Keep	Keep	Keep	Keep	Keep
C-2 or C-3 Date after Assembly Date	Keep	Keep	Keep	Keep	Keep	Keep

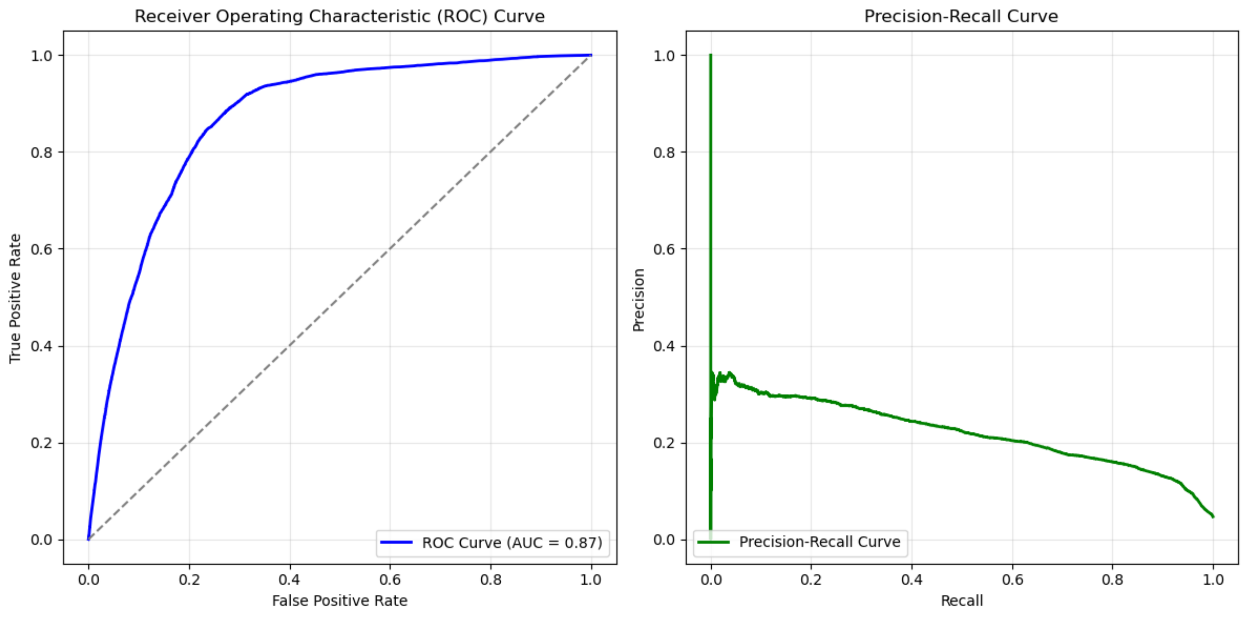
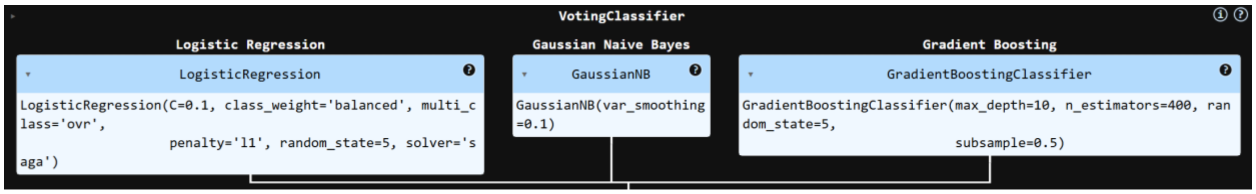


Figure 8. *ROC Curve and AUC & Precision-Recall Curve (Binary Model: Voting Classifiers)*



```
LogisticRegression(C=0.1, class_weight='balanced', multi_c
lass='ovr',
                   penalty='l1', random_state=5, solver='s
aga')
```

```
GaussianNB(var_smoothing
=0.1)
```

```
GradientBoostingClassifier(max_depth=10, n_estimators=400, ran
dom_state=5,
                           subsample=0.5)
```

Figure 9. *Voting Classifier for the Binary Prediction*

Section G: Cross-Validation

Remark:

- columns_to_scale = ['Age at Injury']
- ordinal_columns = ['delay_days_category', 'IME-4 Count Category', 'Wage Category', 'missing_info_category', 'Age Cluster']
- one_hot_columns = ['Medical Fee Region', 'Missing_Dates', 'Mapped Industry Code Description', 'Mapped WCIO Nature of Injury Description', 'Mapped WCIO Cause of Injury Description', 'Mapped WCIO Part Of Body Description','Mapped Carrier Type']

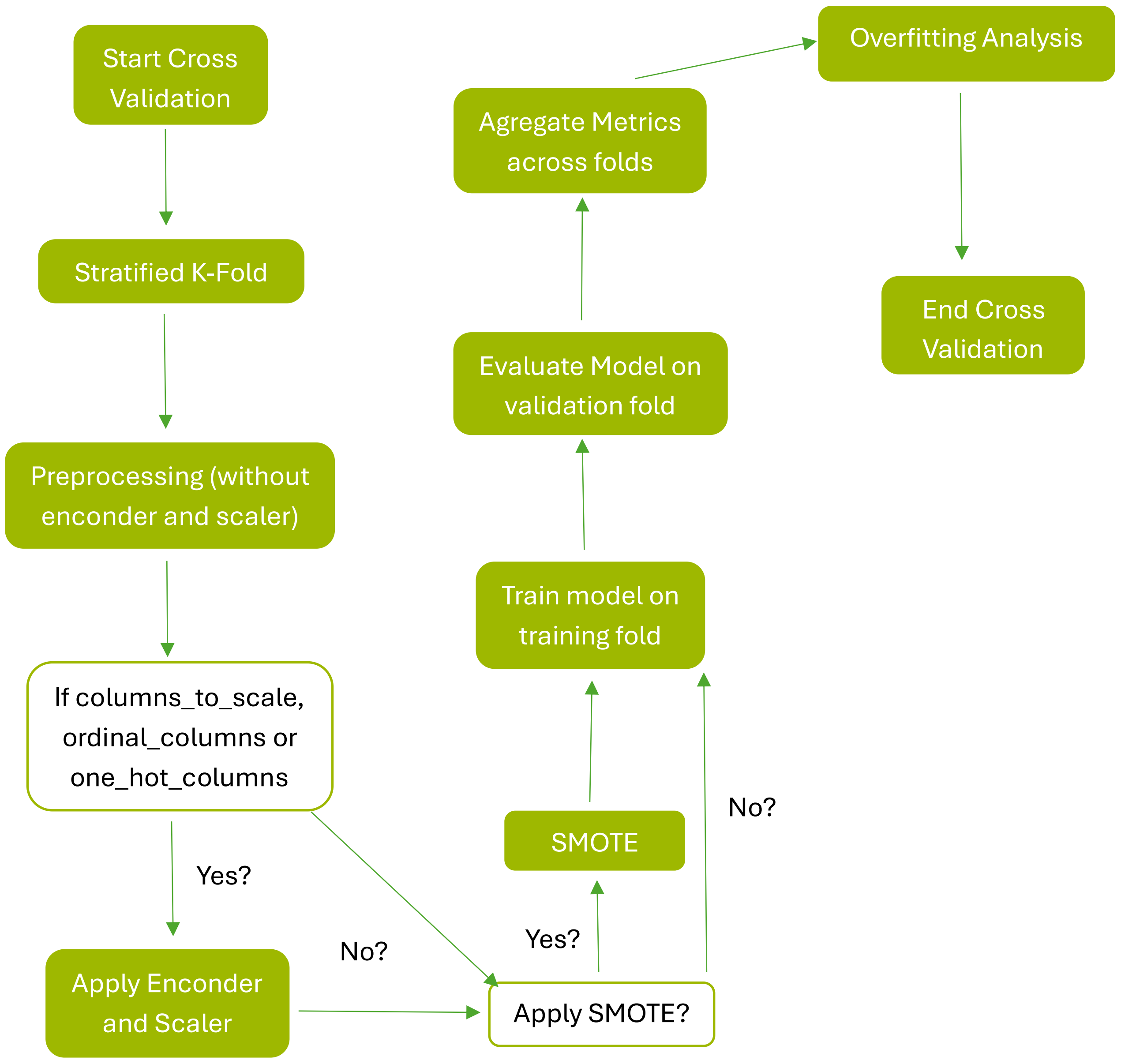

Start Cross Validation
Stratified K-Fold
Preprocessing (without enconder and scaler)
If columns_to_scale, ordinal_columns or one_hot_columns
Yes?
Apply Enconder and Scaler
No?
Apply SMOTE?
Yes?
SMOTE
No?
Train model on training fold
Evaluate Model on validation fold
Agregate Metrics across folds
Overfitting Analysis
End Cross Validation

Figure 10. *Cross Validation Schema (used to evaluate the model's performance)*

Section H: Conclusion

Table 16. *Comparison of LightGBM, CatBoost, and XGBoost. From (Ke et al., 2017; Prokhorenkova et al., 2018; Chen & Guestrin, 2016)*

Aspect	CatBoost	LightGBM	XGBoost
Tree Construction	Symmetric trees: Nodes in the same level have the same splitting condition.	Leat-wise growth: Only one leaf grows per iteration based on the best splitting condition.	Depth-wise growth: Standard decision tree with balanced depth.
Categorical Handling	Ordered encoding: Avoids target leakage by using a permutation-driven method.	Bin/bucket: Groups values into bins for efficient processing.	No predefined method: Requires manual preprocessing or encoding.
Sampling Method	MVS (Minimal Variance Sampling) or Uniform sampling.	GOSS (Gradient-Based One-Side Sampling): Focuses on larger gradients.	Bootstrapped: Samples data points with replacement.

Table 17. *Overview of Gradient Boosting and XGBoost. From (Chen & Guestrin, 2016; Li, n.d.)*

Aspect	Gradient Boosting	XGBoost
Objective Function	Minimizes a loss function without built-in regularization.	Minimizes a regularized objective function combining L1/L2 penalties to avoid overfitting.
Handling Missing Data	Requires explicit preprocessing for missing data.	Efficient handling of missing data without explicit preprocessing.
Cross-Validation	Requires manual implementation for cross-validation.	Built-in cross-validation capability at each iteration to optimize the number of iterations.
Tree Pruning	Splitting stops based on predefined criteria like maximum depth	Uses a "depth-first" tree pruning approach, pruning trees in a backward direction.
System Optimization	No advanced system-level optimizations for speed.	Parallelized tree building, out-of-core computing, and hardware optimizations for speed.
Speed and Efficiency	Slower due to sequential tree building.	Faster due to parallelized tree building and optimized computations.

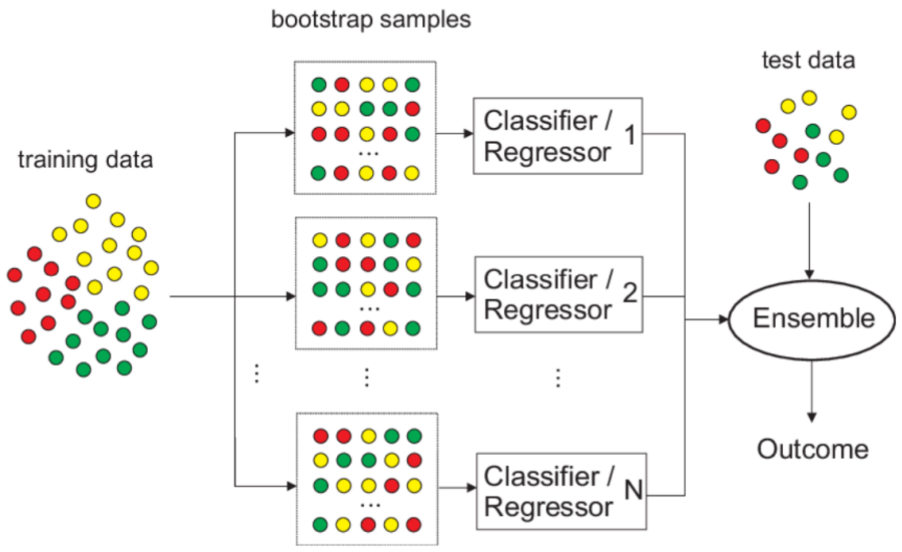


Figure 11. *Illustration of Bagging. From (Ascenso et al., 2022)*

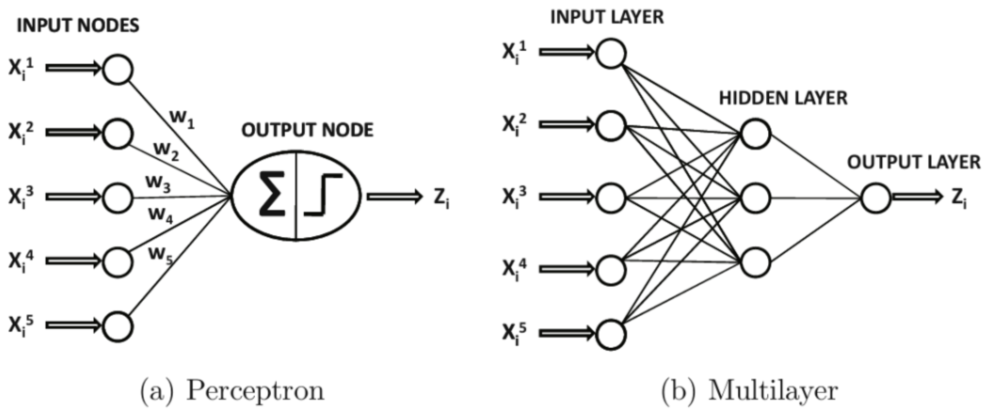


Figure 12. *Single and multilayer neural networks. From (Aggarwal, C.C, 2015)*

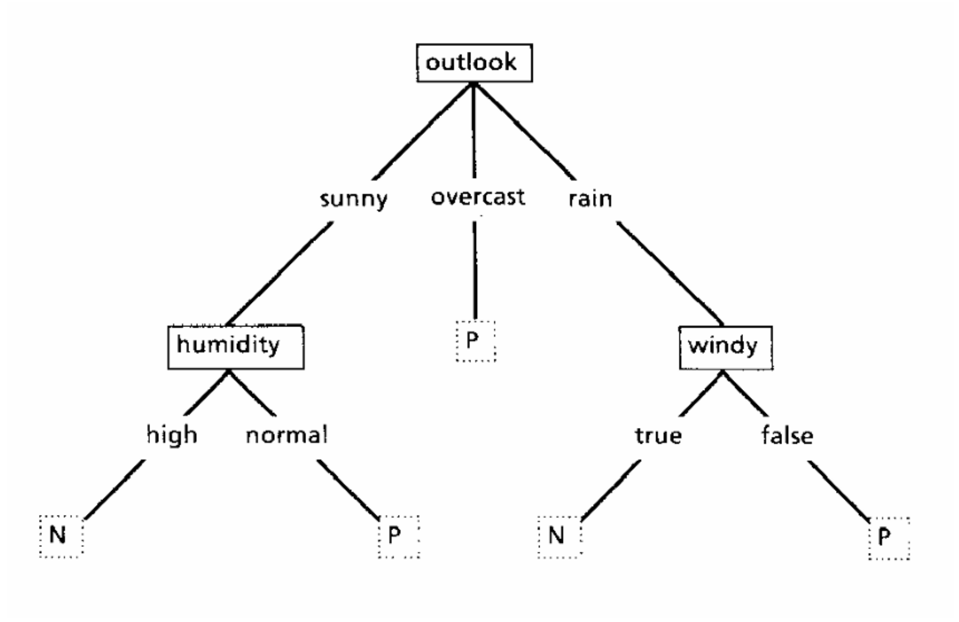


Figure 13. *Example of a simple decision tree. From (Quinlan, J. R. , 1986)*

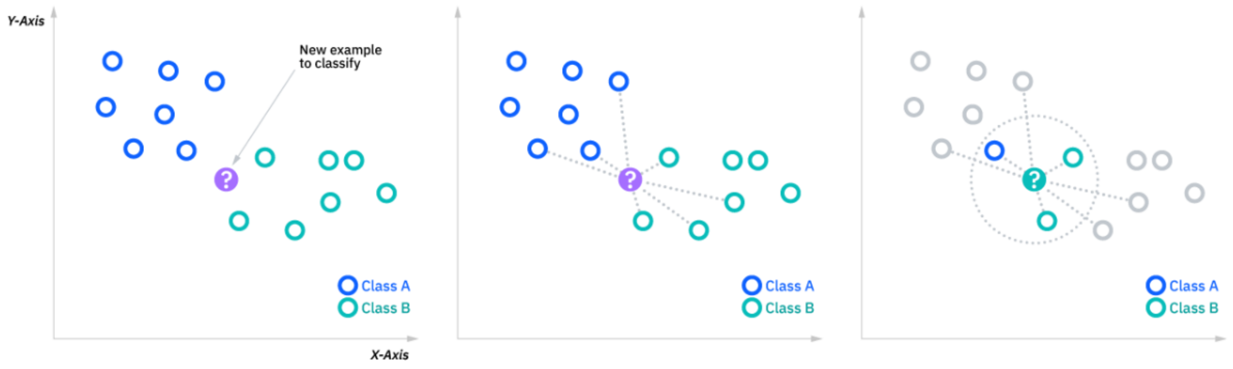


Figure 14. *K Nearest Neighbors Example. From (IBM)*