NOVA Information Management School
Master in Data Science and Advanced Analytics
Machine Learning

To Grant or not Grant: Deciding on Compensation Benefits

DATA SCIENTIST MANAGER: ANTÓNIO OLIVEIRA,20211595
DATA SCIENTIST SENIOR: TOMÁS RIBEIRO, 20240526
DATA SCIENTIST JUNIOR: GONÇALO PACHECO, 20240695
DATA ANALYST SENIOR: GONÇALO CUSTÓDIO, 20211643
DATA ANALYST JUNIOR: ANA CALEIRO, 20240696

Fall/Spring Semester 2024/2025

Table of contents

Glossary

Acronym	Full Name
WCB	New York Workers' Compensation Board
LR	Logistic Regression
DT	Decision Tree Classifier
NB	Naive Bayes
KNN	K-Nearest Neighbors Classifier
RF	Random Forest Classifier
GBC	Gradient Boosting Classifier
XGBoost	Extreme Gradient Boosting
LGBM	Light Gradient Boosting Machine
HGBoost	Histogram-based Gradient Boosting
RFE	Recursive Feature Elimination
PCA	Principal Component Analysis
SVM	Support Vector Machines
K-Folds	Cross-validation technique that splits data into *K* subsets.

Table 1 - Acronym and their Full Name

Term	Description
F1-Score (Macro)	A harmonic mean of Precision and Recall averaged across all classes.
Precision	Ratio of correctly predicted positive observations to total positives.
Recall	Ability of a model to capture all relevant instances in a class.
Accuracy	Overall correctness of a model's predictions.
Grid Search	Exhaustive search for optimal hyperparameters using all combinations of a parameter grid.
Random Search	Random sampling of hyperparameter combinations for optimization.
Stratified Cross-Validation	Cross-validation technique maintaining class distribution across training and validation splits.
Holdout Method	Model assessment method splitting data into training and validation sets.
Ball Tree	A nearest-neighbor search algorithm for imputation.
Oversampling/Undersampling	Techniques to balance class distribution in imbalanced datasets.
Pandas Profiling	Automated library for generating data summaries and visualizations.
RobustScaler	Scaling technique robust against outliers.
Streamlit	Python library for building and deploying interactive web applications.

Table 2 - Term and their Description

1. Abstract

This report focuses on developing a model to classify claims submitted to the New York Workers' Compensation Board (WCB) based on the Claim Injury Type. Using claims data from 2020 to 2022, the project aimed to automate and streamline the decision-making process. Data preparation involved handling missing values, detecting outliers, and implementing feature engineering techniques that were used to improve data and ,consequently, model quality and reliability. Several machine learning methods were explored, including ,Logistic Regression (LR), Decision Tree Classifier (DT), Naive Bayes (NB), K-Nearest Neighbor Classifier (KNN), Random Forest Classifier (RF) and Extreme Gradient Boosting Classifier (XGBoost).Model performance was evaluated using F1-Score, Precision, and Recall, with a particular emphasis on F1-Score. Initially, the Holdout method was applied, identifying 3 top-performing models: XGBoost, Random Forest (RF), and LightGBM (LGBM). After, Stratified Cross-Validation was employed in these models, revealing XGBoost as the best-performing one. It achieved an F1-Score of 0.47, a Precision of 0.55, and a Recall of 0.45, aligning with expectations based on prior research. This study offers a scalable and efficient approach to managing insurance claims, effectively reducing manual effort. Future work could explore the integration of more complex and recent models, as well as improving the interface to include predictions for Agreement Reached, and more.

Keywords: Machine Learning, Multiclass Classification, Gradient Boosting, Predictive Modelling, Holdout Method, Stratified Cross-Validation, Feature Selection, F1-Score, Precision, Recall.

2. Introduction

Accidents can happen to anyone at any time, and if they occur during work hours, employees will seek compensation to address their injuries. Manually reviewing all claims is a tedious and time-consuming process. As a result, the insurance industry, both private and public, is increasingly adopting automated methods to deal with it.

In line with this trend, this project aims to assist the New York Workers' Compensation Board (WCB) in automating the decision-making process for classifying claims by predicting the Claim Injury Type. The task involves creating and optimizing supervised machine learning models using historical data from claims assembled between 2020 and 2022.

Since this is multiclass classification problem we will made use of various classification algorithms to fulfill our goal, such as Logistic Regression (LR), Decision Tree Classifier (DT), Naive Bayes (NB), K-Nearest Neighbor Classifier (KNN), Random Forest Classifier (RF) and Extreme Gradient Boosting Classifier (XGBoost) and more.

Prior Research of related work and expectations for our project:

Through our research we encountered multiple papers that addressed different challenges in the insurance industry using machine learning. We focused more on the ones most aligned with our objectives, such as trying to classify the insurance risk and decide whether insurance should be provided to employees or not. The researchers experimented with multiple models, including Linear Regression, Decision Trees, Random Forest, XGBoost, as well different methods of evaluation like the confusion matrix, F1-Score, Recall and Precision.

Almost all of the studies identified XGBoost algorithm to be the one that demonstrated the higher ability to predict if a worker is going to receive or not receive compensation or a claim status. Even though nearly all of the referenced works are for binary classification problems, one study addressed a multiclass classification one, further reinforcing the predominance of the XGBoost. Consequently, in our own predictions, we also expect this model to be the one that performs higher in the different metrics of evaluation. To see a resume of the referenced works and their findings check: (see table 3 in Appendix)

3. Data Exploration and Pre-Processing

3.1. Data Understanding/Description

The dataset used in this study, provided by the New York Workers' Compensation Board (WCB), consisted of 593 471 rows and 32 columns. Among these variables, there were 11 numerical features, all of which were floats, while the remaining 21 variables were categorical. The target variable, "Claim Injury Type," consists of 7 classes representing the status of the claim, making this a multiclass problem. These classes are highly imbalanced. The classes "2. NON-COMP" and "4. TEMPORARY" have the largest number of samples, with 291 078 and 148 507 instances, respectively.

On the other hand, the classes “8. DEATH” and “7. PTD” have only 470 and 97 samples, respectively. (see Fig 1 in the Appendix)

By applying the Profile Report from the Pandas-Profiling library to the training and test datasets, we identified several issues that required attention. The first major observation was that the columns “WCB Decision” and “Agreement Reached” were missing in the test dataset, and the “OIICS Nature of Injury Description” column was entirely null. These columns were dropped immediately before proceeding with further analysis. Additionally, we found some minor inconsistencies such as the carrier type having 8 unique values in the training data, when only 7 were present in the test data. From the histograms of numerical and categorical variables, we observed key insights like:

- Extreme outliers were present in “Age at Injury” (e.g., age > 100) and “Average Weekly Wage”, along with invalid values in “Birth Year.”
- Most distributions were highly skewed, except “Age at Injury,” which was nearly normal.

The bivariate analysis revealed no strongly positive relationships between variables. However, “Age at Injury” and “Birth Year” were highly negatively correlated (-0.98), which is expected as the age can be computed from the birth year. No significant linear relationships were identified. For the target variable analysis against all other variables, male claims dominate most injury types, particularly temporary and non-compensable ones, while claims without attorney involvement (N) are largely non-compensable or temporary, which makes sense. (see EDA figures for more insights in Appendix)

3.2. Proposed Architecture of the Claim Injury type Prediction Model

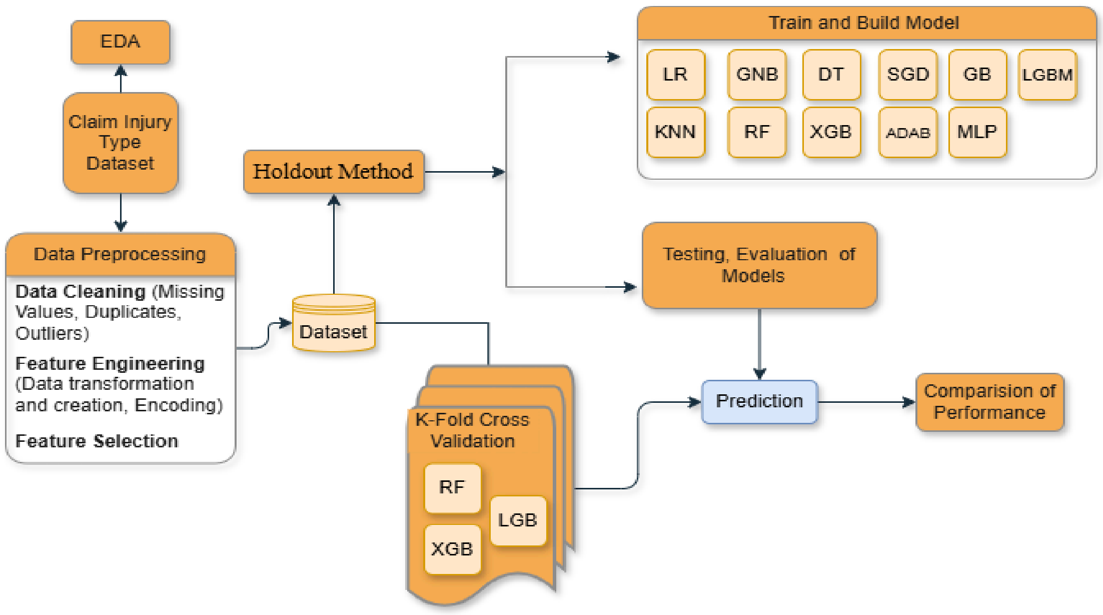

EDA
Claim Injury Type Dataset
Data Preprocessing
Data Cleaning (Missing Values, Duplicates, Outliers)
Feature Engineering (Data transformation and creation, Encoding)
Feature Selection
Dataset
Holdout Method
Train and Build Model
LR
GNB
DT
SGD
GB
LGBM
KNN
RF
XGB
ADAB
MLP
Testing, Evaluation of Models
K-Fold Cross Validation
RF
LGB
XGB
Prediction
Comparision of Performance

Fig 9 - Architecture of the Claim Injury type Prediction Model

After gaining a solid understanding of our data, we plan to test different models and features using the Holdout Method. This approach allows us to avoid performing feature selection during K-fold Cross-Validation, which takes a lot of time to process.

3.3. Data Cleaning and Preprocessing

We started to clean the data by handling missing values on the target variable and applying manual label mapping for consistency in the modeling as well as dropping duplicates in the dataset, and grouping classes. For example, “Carrier Type”, where categories labeled '5' were combined into '5. SPECIAL FUND,', addressing the incoherent mentioned before, and two less frequent genders were merged into one category.

<u>Feature Creation</u>

To reduce dimensionality and/or extract more meaningful insights we created new variables:Extracted year, month, day, and weekday from date columns.

Calculated time intervals between key events (Accident to Assembly, C-2 Receipt to Assembly, and Assembly to C-2 Receipt) to improve target predictions, assuming the timeline based : Accident Date → C-2 Date → Assembly Date

Combine WCIO codes into one column, transforming them into integers by filling missing values with 0 and converting negative codes to their absolute values, treating them as categorical representations.

Although they are now numeric, they are not actual numbers and just a representation of their descriptions.

Added new variables: Insurance to identify carrier names starting with "ins," Zip Code Valid to assess zip code validity, and grouped ages into categories.

3.3.1. Train-Test Split

Before Proceeding with further Feature Engineering we split the data through the Holdout Method. We applied a split of 20% of the dataset for the validation and the remainder will be for training. This step is important to avoid data leakage.

3.3.2. More Feature Engineering - Encoding

We applied different encoding methods to our features based on their characteristics. For variables with a considerable number of unique values, we used Count Encoding. An exception was the variable “Medical Fee Region,” which has only five unique values, however, due to the lack of a clear ordinal relationship, we still applied Count Encoding. Another exception was the variable “Carrier Type,” which was encoded using both Count Encoding and One-Hot Encoding. The best approach will be selected during feature selection. For variables with a small number of unique values, we used One-Hot Encoding, while for binary features with only two categories, we applied Binary Encoding. An exception was the variable “Alternative Dispute Resolution,” which has three categories. In this case, we grouped “Y” and “U” together as 1. (see Table 4 in Appendix)

3.3.3. Missing Values

The treatment of missing values involved 4 approaches: creating new features based on the existence of missing values, filling missing values with constants, using statistical methods, and applying predictive models. Using the first approach, we created a new binary variable, “C-3 Date,” where it is assigned 0 if the C-3 Date is missing and 1 otherwise. Similarly, we created another binary variable, “First Hearing Date,” following the same logic.

For filling missing values with constants, we filled the variable “IME-4 Count” with 0, as a missing value indicates that no form was created. For the variable “Industry Code,” we filled missing values with 0, assuming a missing instance represents an unknown code.

For statistical methods, we handled date-related columns such as year, month, and day of “Accident Date” and “C-2 Date” by imputing missing values with the median. We then recomputed the full date and filled in missing day-of-week information, along with new variables starting with “Time Between.”

In terms of predictive methods, for the variable “Birth Year,” we created a mask to filter observations where both Age at Injury and Accident Date Year were present, and Birth Year was either missing or zero. Since Age at Injury and Accident Date Year are necessary to calculate Birth Year, ensuring these two variables were not missing was essential. Additionally, we recomputed the Birth Year when it was 0, as a value of zero is unrealistic.

Finally, for the variable “Average Weekly Wage” we filled the missing values using the Ball Tree as an imputation technique. We initially considered scaling all numeric and count-encoded features before filling missing values, as doing so afterward could result in inconsistencies. However, after evaluating models with scaling applied both before and after imputation, we found that the performance differences were insignificant. Therefore, we decided to proceed with scaling the features first.

3.3.5. Outliers

To detect outliers, we used boxplots and the Interquartile Range (IQR) method with a small threshold of 0.001. This allowed us to analyze more features with potential outliers and manually check the upper and lower bounds before making the final decision. For instance, we handled the "Age at Injury" feature, knowing that people of a certain age are probably retired, so we set the upper bound at 88.5 years, as identified by the IQR. We also applied the upper bound of 2017 for the "Accident Date Year" and "C-2 Date Year", and set the lower bound for "Birth Year" at 1932.5.

Initially, we experimented with a square root transformation for the "Average Weekly Wage" and a log transformation for the "IME-4 Count" to see if the transformations would reduce outliers or skewness. Therefore, these features improved the performance of our models, being part of the best one.

4. Multiclass Classification

4.1. Feature Selection Strategy

To see our feature selection approach in more depth please go to: (Table 5 in Appendix).

We experimented with Recursive Feature Elimination (RFE) using both Random Forest and Logistic Regression models. Although well-implemented, RFE proved computationally very expensive and was ultimately excluded. Initially, we applied low thresholds, and Stratified K-fold Cross-Validation revealed that the optimal model performed better when utilizing all features. However, later in the project, to optimize both efficiency and performance, we retained only the key numerical features such as “Age at Injury,” “Average Weekly Wage,” “IME-4 Count,” and time-based variables. Similarly, important categorical features included “Carrier Name Enc,” time-related categorical variables, and others. On the other hand, features like “Carrier Type” categories, “Zip Code Valid,” and “Gender_U/X” were considered the most redundant and excluded. It is important to note that not all feature selection methods were given equal importance. For instance, methods like variance thresholds did not eliminate any features so we gave less relevancy. Instead, Lasso selected features that served as our main guide, complemented by the Extra Trees Classifier, which reinforced the importance of almost the same variables. To see in more detail check : (Table 6 and Table 7 in Appendix)

4.2. Model Assessment Strategy

Our model Assessment Strategy consisted, as we mentioned before, in using the Holdout Method to gain a broad idea about the performance of different models and different features so we could avoid performing feature selection during K-fold Cross-Validation as well as all the models, since it takes a lot of time to process.

For the Holdout Method we split our dataset in 20% for the validation phase and the remaining for the training. We tested multiple models, including LR, as our base model, SGD, RF, AdaBoost, GBC, DT, XGBoost, MLP, GNB, LGBM and KNN. For the K-fold Cross-Validation, we selected only the models XGBoost, RF and LGBM. We will explain later in the report the logic and metrics used behind these choices.

The question that remains is why we chose to use Cross-Validation, specifically Stratified Cross-Validation, when we had already trained our models using the Holdout Method. A fundamental assumption in machine learning is that having more data for training generally leads to better model performance. Cross-validation allows us to use all available data, which helps in reducing overfitting and improving generalization across our models. In essence, this means we can say, in a robust way, that cross-validation allows us to make better use of our data. We specifically chose Stratified Cross-Validation because it is particularly useful when dealing with imbalanced class distributions, which is the case of our Target Variable. To apply this method we had to create a pipeline that incorporated all the important steps for the final prediction. Initially, we experimented with the default nº of folds (5) and then proceeded to use 10 folds, the most recommended nº in the literature. We also tried with 15 and 20 folds to see if there were bigger improvements, which were not verified. Resampling techniques like the Random Over and Under Sampler were used to try to balance the data in parallel with the Stratified K-Folds Method. However they were discarded later due to redundancy in the final score.

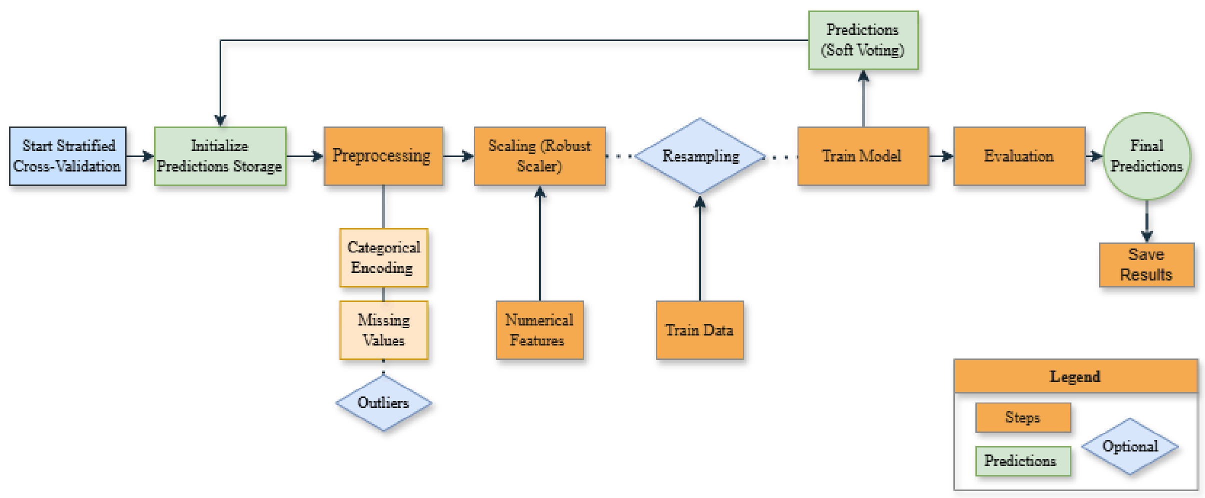


Fig 10 - Pipeline of Stratified Cross-Validation

4.3. Comparison of Metrics Used and Performance

The performance of the models was evaluated using key metrics such as F1-Score (macro), Precision and Recall for both the training and validation datasets. Given the imbalanced nature of the target variable, F1-Score (macro) was prioritized, as it balances Precision and Recall across all classes, providing a more accurate measure for minority class performance. For the same reason, the imbalanced nature of the target variable, we did not use Accuracy as a performance metric. This is because Accuracy can be misleading when one class has significantly more samples than the others. For example, even a high accuracy score does not mean all classes are predicted, specially the minority ones.

LR, serving as the baseline model, provided a benchmark but performed poorly, particularly on the validation set, yielding low Precision and Recall metrics. SGD showed minor improvement over LR but still failed to achieve satisfactory results. DT and RF demonstrated near-perfect performance on the training data but significantly overfitted, as reflected by their much lower validation scores.

Ensemble methods, including Gradient Boosting Classifier (GBoost), XGBoost, and LGBM, achieved a better balance between training and validation performance. Among these, XGBoost consistently delivered the highest validation F1-Score, showcasing its ability to manage class imbalance effectively and generalize better to unseen data. MLP and KNN struggled with generalization and underperformed compared to the boosting methods.

It is worth nothing that tree-based models like DT and RF were disregarded for final training due to overfitting issues. For Stratified K-Fold Cross-Validation, the models retained were XGBoost, LGBM, and RF. While GBoost and HGBoost demonstrated promising results, faster and more lightweight models like LGBM and XGBoost were prioritized for cross-validation.

	Data	Macro F1	Precision	Recall
1st	Train	XGB	XGB	XGB
	VaL	XGB	GBoost	XGB
2nd	Train	GBoost	GBoost	HGBoost
	VaL	GBoost	XGB	GBoost
3rd	Train	HGBoost	KNN	GBoost
	VaL	RF	RF	HGBoost

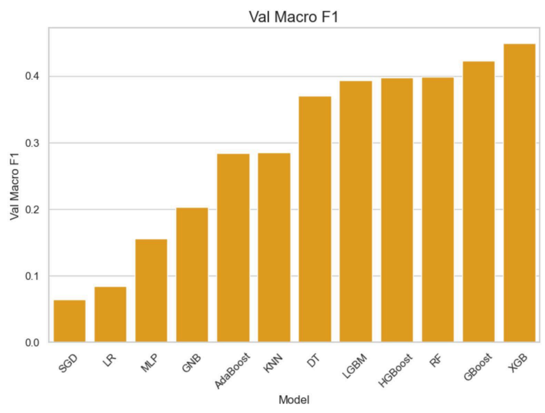


Table 7 - Performance of Results and Fig 11 - Macro F1 score for Validation dataset, both for the HoldOut Method

In conclusion, XGBoost emerged as the best-performing model, achieving the highest scores across all metrics during validation and Stratified K-Fold Cross-Validation. The final model utilized 10 folds, without resampling techniques, and incorporated the optimal parameters derived from the hyperparameter tuning process described on model optimization. This approach achieved a Macro F1-Score of 0.50 on Kaggle, validating the model's effectiveness for the multiclass classification problem. To visualize our results more in depth check: (Performance of the models in Appendix)

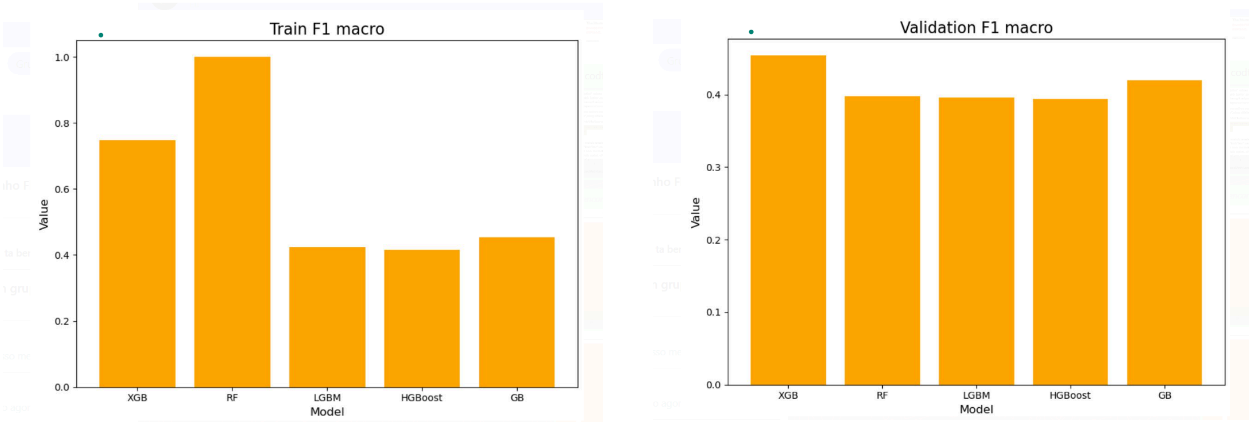


Fig 12 - Macro F1 score for the used in Stratified Cross-Validation

4.4. Model Optimization

To enhance the performance of our top-performing models, we employed hyperparameter tuning using Grid Search and Random Search methods. This process allowed us to fine-tune parameters for Gradient Boosting (GBC), XGBoost, and LightGBM (LGBM), ensuring optimal results. The selection of parameters was guided by the F1-Score (macro) metric due to the imbalanced nature of the target classes.

For XGBoost, the parameters fine-tuned included n_estimators, max_depth, learning_rate, subsample, and colsample_bytree. Random Search, with a 5-iteration limit and 2-fold cross-validation, was used to efficiently explore these parameter combinations.

For Gradient Boosting (GBC), the parameters tuned included n_estimators, learning_rate, max_depth, min_samples_split , and min_samples_leaf.

For LightGBM (LGBM), the tuning process focused on n_estimators, learning_rate, and num_leaves .

The optimization outcomes revealed that XGBoost achieved the best performance, with its optimized parameters (learning_rate: 0.4, n_estimators = 200) for a 5-fold cross validation, leading to a significant improvement in the F1-Score (macro). Both Gradient Boosting and LightGBM also demonstrated performance gains but did not surpass XGBoost. Additionally, Random Search proved to be more efficient than Grid Search, reducing computational time while delivering comparable results.

5. Open-Ended Section

For the open-ended section, our team thought it was interesting to develop an analytics interface that provides predictions based on new user inputs. Our main goal is to deliver a quick, intuitive experience, allowing clients, specifically, the New York Workers' Compensation Board (WCB), to access and understand the outputs generated by our model, without the need of reaching out to us. To use the application, the WCB simply completes a short form on the "Inputs and Predictions" page and clicks on the "Predict" button to receive the predicted type of injury for the worker. A secondary goal of this app is to provide user-friendly and easily interpretable results. We achieve this by allowing users to explore interactive visualizations that make the insights more accessible and engaging. Additionally, we designed the app with scalability in mind, meaning it can be adapted by the clients will to include more predictive options. This will ensure that the app remains relevant as the needs of the WCB are involved. To see the app click on this link:

https://24nexus-analystics.streamlit.app/

5.1. Structure of the app and actions taken

We built, hosted and deployed this app with a streamlit library of python and their respective cloud. The library is very intuitive, allowing to create a engageable interface with few lines of code. However, we encountered some challenges, specially finding a way to encode our users inputs so the model could read them and also, we had to install Git Large File Storage so the cloud of streamlit could find our train data *csv* file.

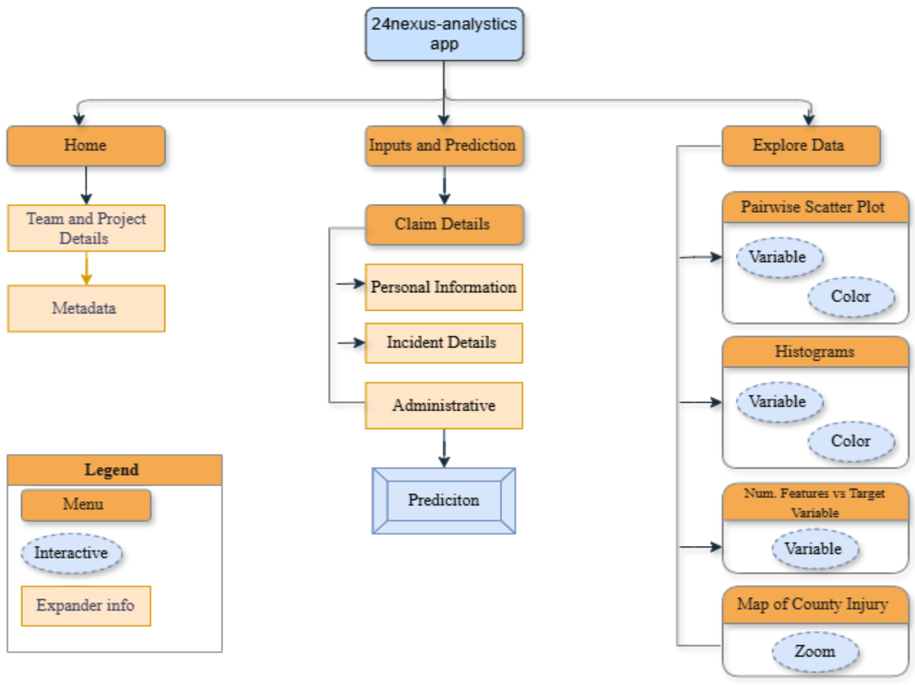


Fig 14 - Structure of the app

Our workflow consisted of 3 main Python files. The app.py serves as the primary file, managing the display of the options menu, visualizations for the "Explore Data" section, and connecting to the prediction functionality. The second file, predict.py, is responsible for creating the sliders and input boxes that the client interacts with when filling out the form in the app. This file also handles collecting user inputs and includes the button to make the predictions, which connects to the preprocessing page. The solution we found to enable predictions, was saving the user's inputs into a CSV file. This file is then read into the preproc.py /(preprocessing page). In this script, we perform basic encoding, load the training data, and replicate the transformations applied during the Holdout Method (training and validation phases). Instead of using test data, we replace it with the user's input data. After applying the necessary transformations, we fit the model as usual and reverse the label encoding to provide a final prediction. The model used for predictions is XGBoost, as it has the best performance. However, the app is designed to support other models if needed.

5.2. Discussion

The app successfully meets its objectives by providing a user-friendly interface that predicts the Claim Injury Type while also allowing users to explore the data interactively. However, there is still room for improvement. For example, since the training dataset contained some unclean features, issues like a "0" appearing in the Birth Year field can be observed in the graphs. Looking ahead, there are opportunities to improve the app further. A valuable addition would be introducing another prediction option where users can determine if a claim resulted in an Agreement Reached or detect potential fraudulent claims.

6. Conclusion

The main goal of this project was to predict the Claim Injury Type while ensuring reliable results and providing an interactive way for the client to explore them. To achieve this, we adopted a detailed approach, thoroughly exploring the data and implementing preprocessing steps that aligned with our understanding. Additionally, we developed a pipeline for stratified cross-validation that integrated these preprocessing steps. For result exploration, we created an interactive interface that met the intended objective.

In summary, we successfully achieved our goals, albeit with some setbacks. After training our models using the Holdout Method and comparing them based on F1 score, accuracy, and precision, we selected the top three models, considering the restrictions previously mentioned. The best-performing model was XGBoost, with an F1 score of 0.47, a precision of 0.55, and a recall of 0.45. Among these metrics, the F1 score was our primary focus. Reinforcing this, XGBoost consistently outperformed the others across all metrics during cross-validation. To achieve these results, we discarded resampling techniques and optimized hyperparameters using Grid Search.

Our findings aligned with prior research, as anticipated, where studies in the same industry often identified XGBoost as a top-performing model. However, our metric scores were slightly lower than those reported in similar multiclass classification studies, which took into account the limitations of our work.

The most significant challenge and limitation of our project was the highly imbalanced target variable (Claim Injury Type), which influenced the generalization accuracy of each model. We applied Stratified Cross-Validation to ensure more reliable results and to maintain similar sample proportions across classes. While we experimented with oversampling and undersampling techniques, these were ultimately discarded as they did not improve the final predictions. In the future, these techniques could and should be explored in greater depth.

Another limitation was the dimensionality of our features. Our feature selection relied heavily on embedded methods and was relatively benevolent. A more critical approach could have helped reduce the number of features further. Future work could reintroduce wrapped methods (RFE) or even explore dimensionality reduction techniques such as Principal Component Analysis (PCA).

Regarding modeling, experimenting with Support Vector Machines (SVM) would be valuable. In a related study on class prediction within the insurance industry, SVM demonstrated strong performance. While we attempted to implement it, the process proved too time-consuming.

Finally, future efforts could also involve more extensive hyperparameter tuning, as well as experimenting with deep learning models featuring multiple hidden layers and activation functions and different methods of assessment of performance, like the ROC curve.

7. Appendix

Reference	Title	Objectives	Algorithms Used	Best Algorithm
Sahai, R. et al. (2023)	Insurance Risk Prediction Using Machine Learning	The study focused on enhancing risk assessment capabilities for life insurance companies using predictive analytics by classifying the insurance risk based on the historical data and propose the appropriate model.	DT, RF and XGBoost	XGBoost
B. K. Saraswat, A. Singhal, S. Agarwal and A. Singh (2023)	Insurance Claim Analysis Using Traditional Machine Learning Algorithms	The sole purpose of choosing this topic is to provide companies with a tool that can help them to do the important work of deciding whether insurance should be provided to employees or not based on some set of information about employees, saving human resources.	LR, KNN, DT, Support Vector Machine (SVM), RF, NB, Gradient Boost (GB), XGboost	XGBoost
Endalew Alamir1 , Teklu Urgessa2 et al. (2021)	Motor Insurance Claim Status Prediction using Machine Learning Techniques	The aims of the study was to build a machine learning model that classifies and make motor insurance claim status prediction in machine learning approach.	RF, SVM	SVM

N. Dhieb, H. Ghazzai, H. Besbes and Y.Massoud (2019)	Extreme Gradient Boosting Machine Learning Algorithm For Safe Auto Insurance Operations	Develop an automated fraud detection approach for auto insurance companies based on extreme gradient boosting algorithm, aka XGBoost.	XGBoost	XGBoost
W. F. Mustika, H. Murfi and Y. Widyaningsih (2019)	Analysis Accuracy of XGBoost Model for Multiclass Classification - A Case Study of Applicant Level Risk Prediction for Life Insurance	Using machine learning model to help classify prospective insurance applicants based on the level of risk quickly.	DT, RF, NB, XGBoost	XGBoost

Table 3 - Research Works

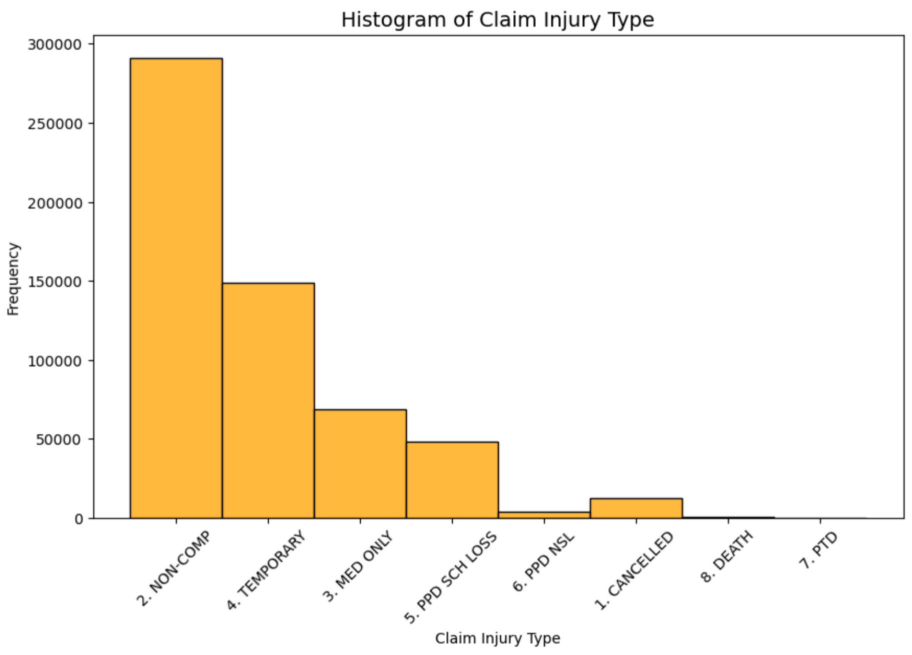


Fig 1 - Claim Injury Type Histogram

Eda Figures

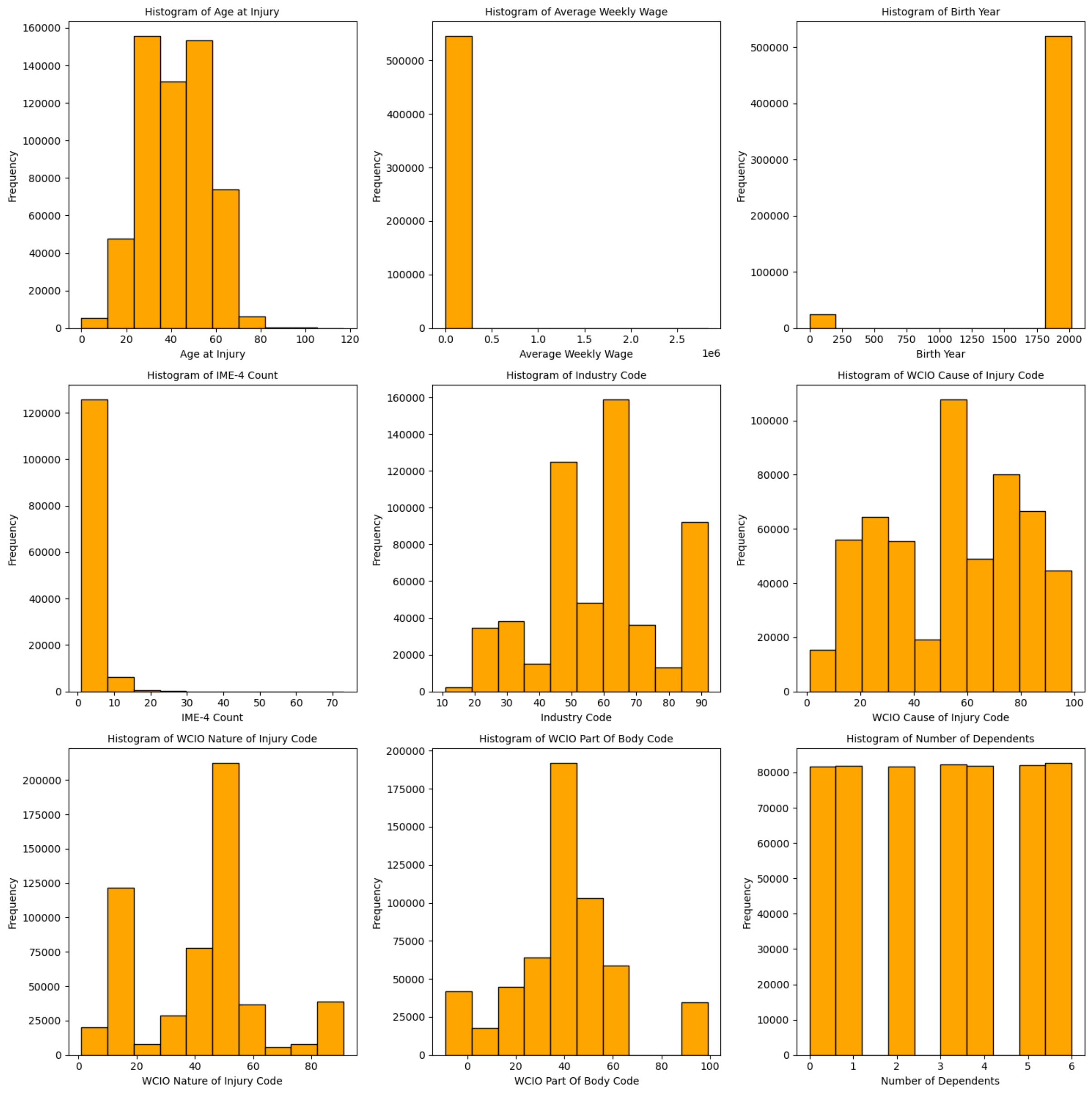


Fig 2 - Histograms of Numerical Variables

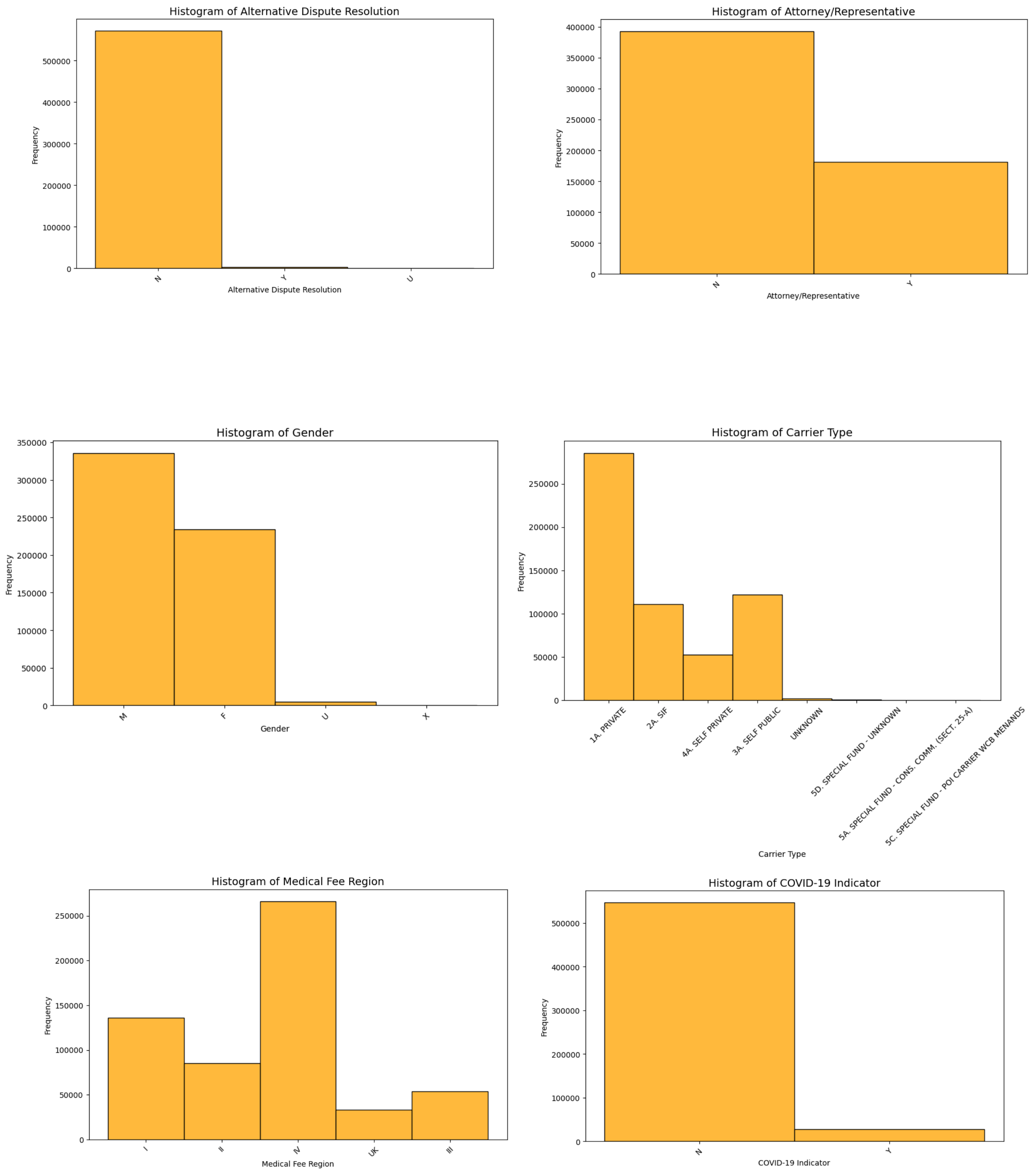


Fig 3 - Bar Charts of Categorical Variables

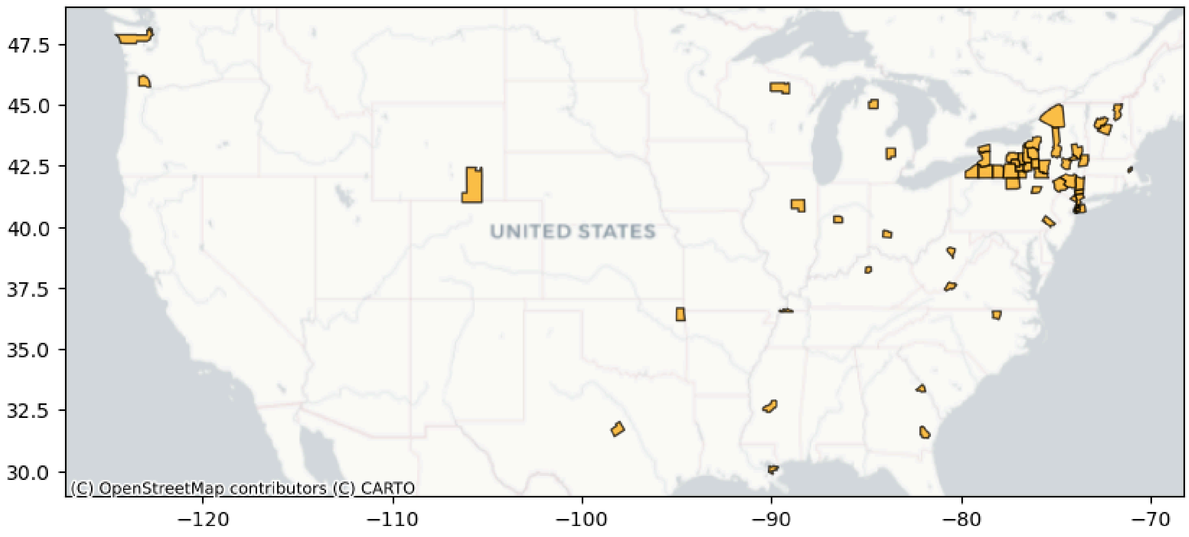


Fig 4 - Map of County Injury

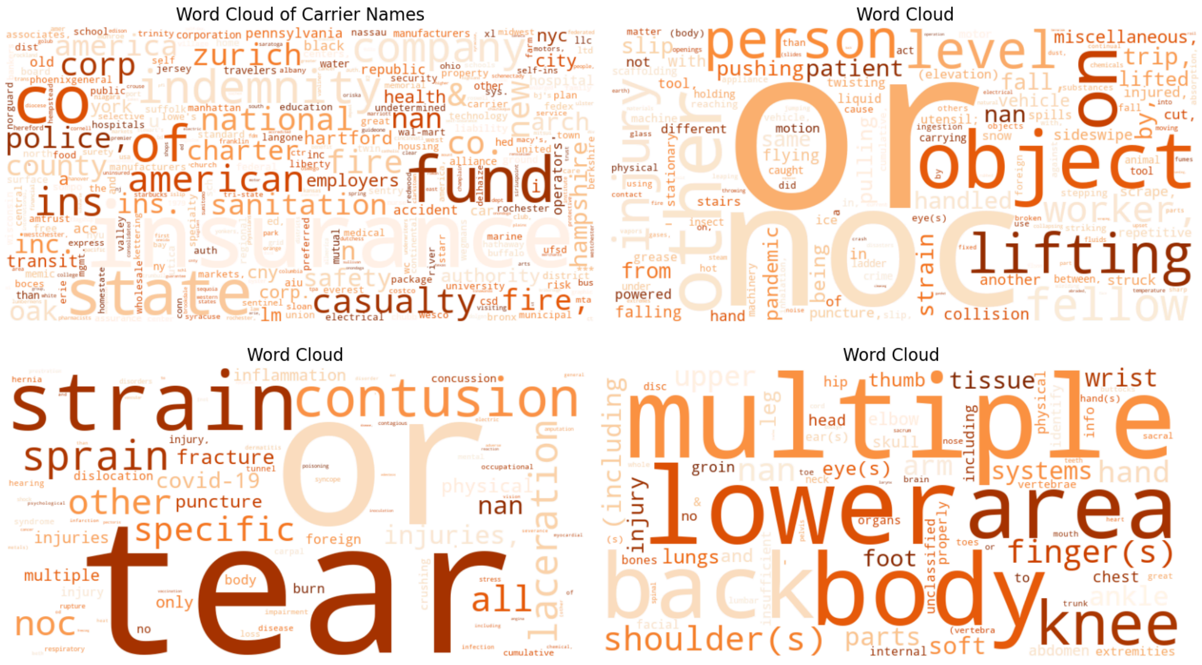


Fig 5 - Word Clouds of the Carrier Names, WCIO Cause of Injury Description, WCIO Nature of Injury Description and WCIO Part of Body Description, respectively

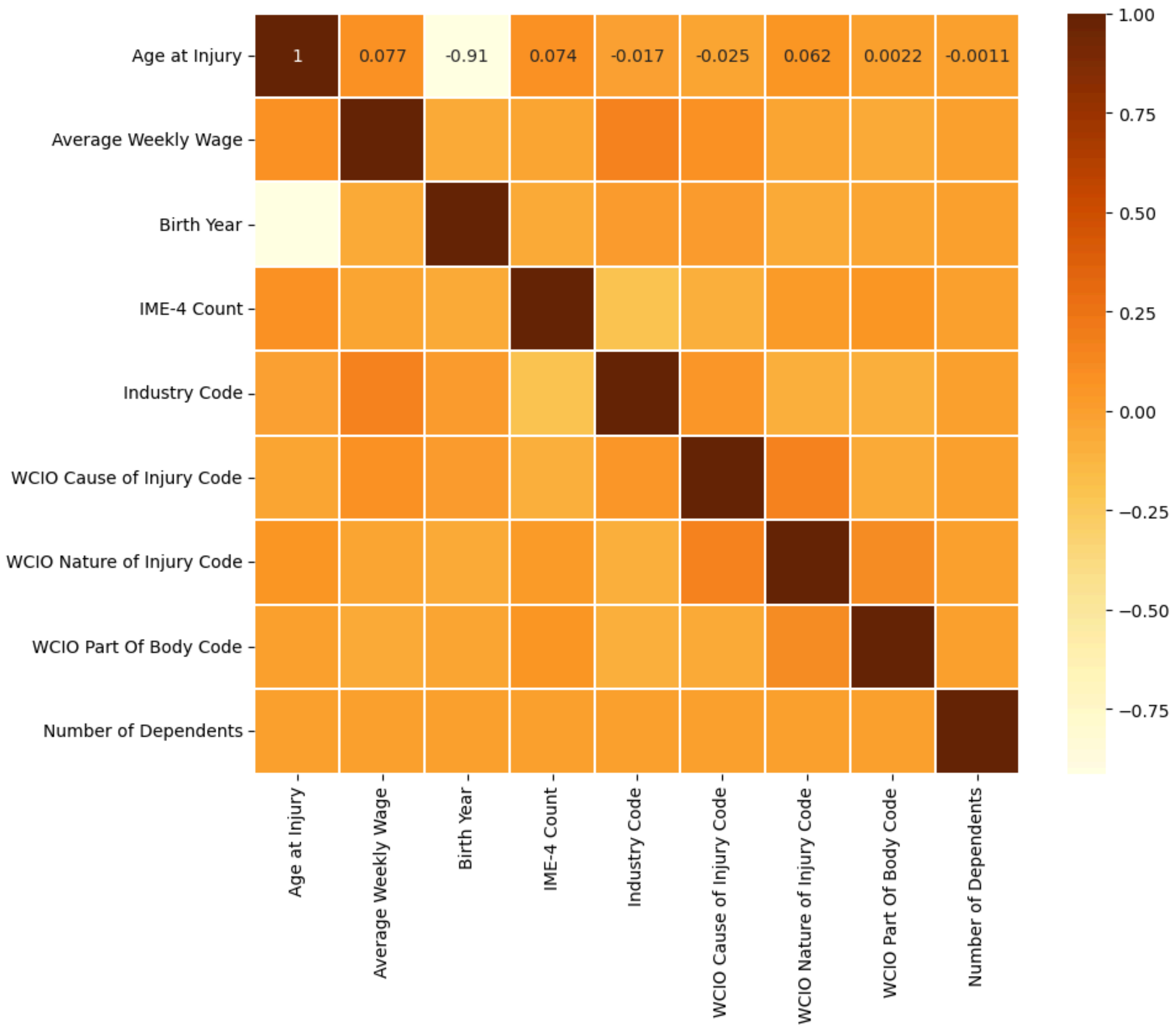


Fig 6 - Spearman Correlation Matrix (Numerical vs Numerical)

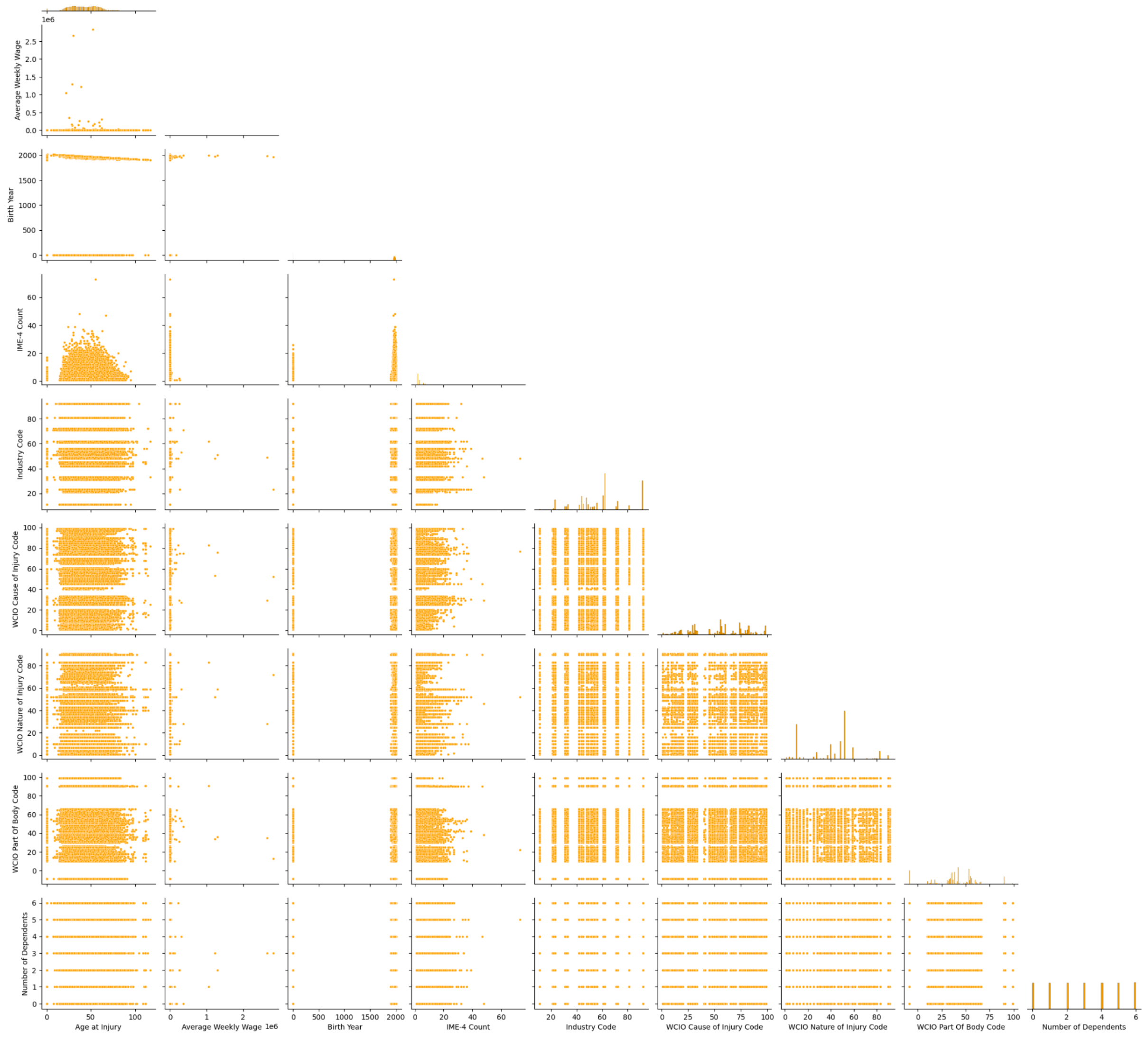


Fig 7 - Pairwise Relationship between Numerical Features

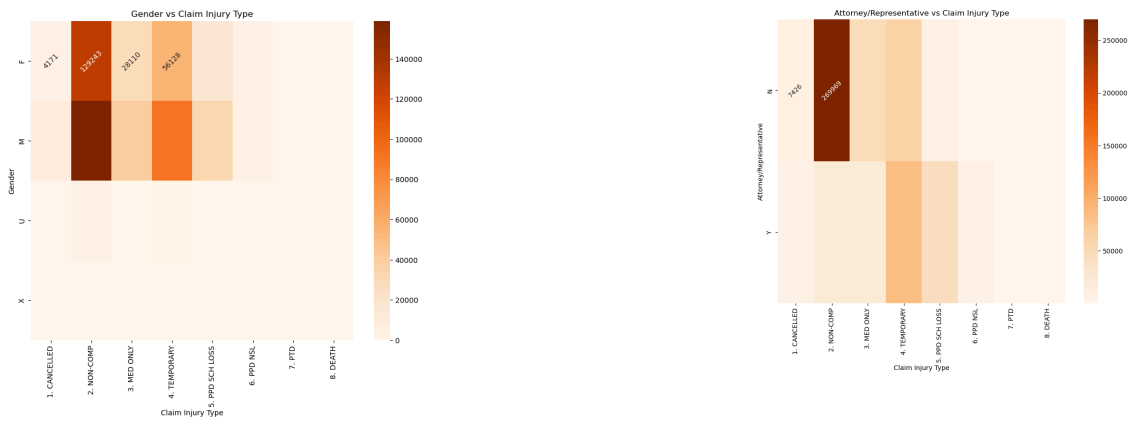


Fig 8 - Categorical vs Claim Injury Type (more were performed on the notebook)

Feature	Encoding Applied	Comments
Age at Injury	No Encoding	Numerical feature; no transformation applied.
Average Weekly Wage	No Encoding	Numerical feature; retained as-is for model input.
Birth Year	No Encoding	Numerical feature; directly used in the model.
IME-4 Count	No Encoding	Numerical feature; log transformations were considered but excluded.
Carrier Name Enc	Count Encoding	Encoded due to a high number of unique values.
Carrier Type Enc	Count Encoding	Encoded for feature selection and better representation.
County of Injury Enc	Count Encoding	Applied Count Encoding to handle many unique values.
District Name Enc	Count Encoding	Transformed for compatibility with model training.
Medical Fee Region Enc	Count Encoding	Applied despite a low number of unique values for consistency.
Industry Sector Enc	Count Encoding	Encoded to simplify categorical representation.
Binary Features (general)	Binary Encoding	Features with two categories were directly encoded as binary variables.
Categorical Features (remaining)	Label or Binary Encoding	Non-count-encoded features were assigned either binary or label encoding.

Table 4 - Encoding Methods

Method		Feature Types	Threshold
Filter-Based	Variance Threshold	Numerical	Var very high
	Correlation	Numerical	0.01
	Chi - Square Test	Categorical	0.05
	Mutual Information	Categorical	0.05
Wrapper	RFE	All	-
Embedded	LASSO	All	0.05
	Tree-Based Feature Importance	All	-

Table 5 - Feature Selection Approach

Numerical Features

Variable	Variance	Correlation	Lasso	ExtraTrees	Decision
Accident Date Day	K	K	D	K	Keep
Accident Date Month	K	K	D	K	Keep
Accident Date Year	K	HC_4	D	D	Discard
Accident to Assembly Time	K	K	K	K	Keep
Accident to C-2 Time	K	K	K	K	Keep
Age at Injury	K	HC_3	K	K	Keep
Assembly Date Day	K	K	D	D	Discard
Assembly Date Month	K	K	D	D	Discard
Assembly Date Year	K	HC_4	K	K	Keep
Assembly to C-2 Time	K	K	K	K	Keep
Average Weekly Wage	K	HC_2	K	K	Keep
Average Weekly Wage Sqrt	K	HC_2	K	K	Keep
Birth Year	K	HC_3	K	K	Keep
C-2 Date Day	K	K	K	K	Keep
C-2 Date Month	K	K	K	K	Keep

C-2 Date Year	K	HC_4	D	K	Keep
IME-4 Count	K	HC_1	K	K	Keep
IME-4 Count Double Log	K	HC_1	D	D	Discard
IME-4 Count Log	K	HC_1	K	K	Keep
Number of Dependents	K	K	D	D	Discard

Table 6 - Feature Selection Decision of Numerical Features

Categorical Variables

Variable	Chi-Squared	MI	Lasso	Extra Trees	Decision
Accident Date Day of Week	D	D	K	K	Keep
Assembly Date Day of Week	D	D	K	K	Keep
Age Group	K	K	K	K	Keep
Alternative Dispute Resolution Bin	K	D	K	D	Keep
Attorney/Representative Bin	K	K	K	K	Keep
Carrier Name Enc	K	K	K	K	Keep
Carrier Type Enc	NA	D	K	D	Keep
Carrier Type_2A. SIF	K	D	D	D	Discard
Carrier Type_3A. SELF PUBLIC	K	D	D	D	Discard
Carrier Type_4A. SELF PRIVATE	K	D	D	D	Discard
Carrier Type_5. SPECIAL FUND OR UNKNOWN	K	D	D	D	Discard
C-2 Date Day of Week	K	D	K	K	Keep
C-3 Date Binary	K	K	K	K	Keep
COVID-19 Indicator Enc	K	D	K	K	Keep
County of Injury Enc	NA	D	K	K	Keep
District Name Enc	NA	D	K	K	Keep
First Hearing Date Binary	K	K	K	K	Keep
Gender Enc	K	D	K	K	Keep
Gender_M	K	D	D	D	Discard

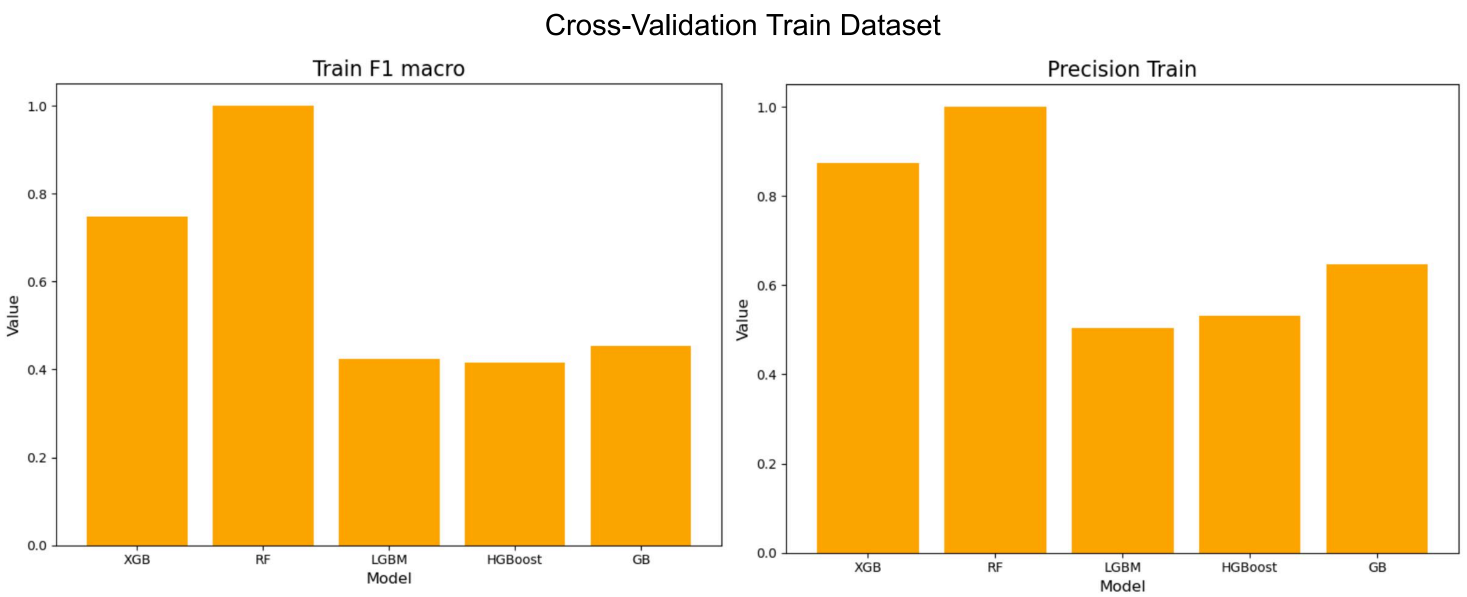
Gender_U/X	K	D	D	D	Discard
Industry Code	K	K	K	K	Keep
Industry Sector Enc	NA	D	K	D	Keep
Insurance	K	D	D	D	Discard
Medical Fee Region Enc	NA	D	K	K	Keep
WCIO Cause of Injury Code	K	K	K	K	Keep
WCIO Codes	K	K	K	K	Keep
WCIO Nature of Injury Code	K	K	K	K	Keep
WCIO Part Of Body Code	K	K	K	K	Keep
Zip Code Valid	D	D	D	D	Discard

Table 7 - Feature Selection Decision of Categorical Features

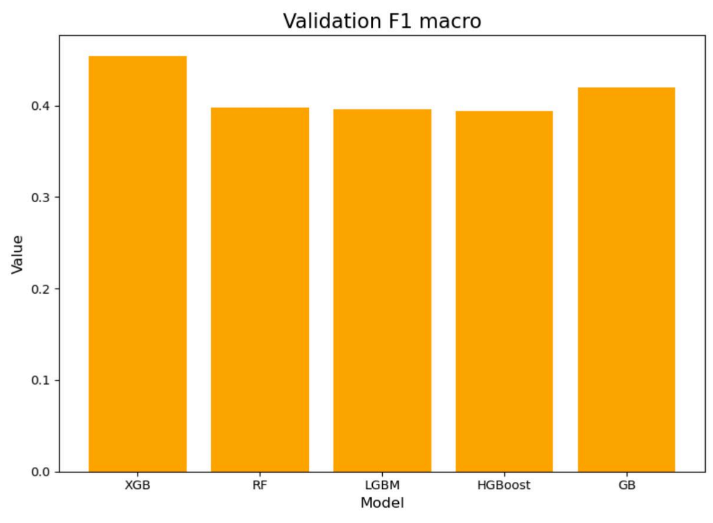
Symbol	Meaning
K	Keep
D	Discard
HC_N	High Correlation Value
NA	Not Applicable

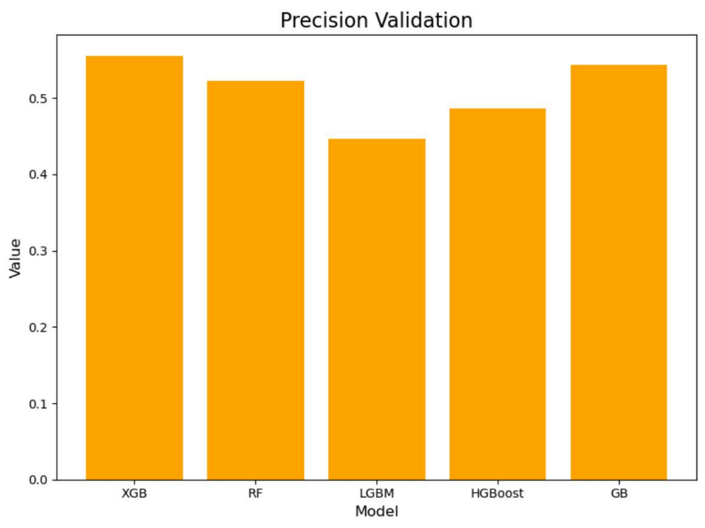
Legend of the symbols used in Feature Selection

Fig 13 - Performance of Models

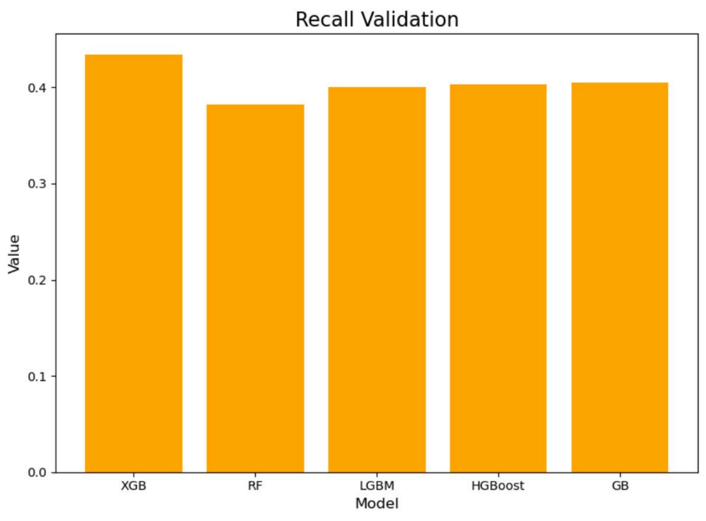


Cross-Validation Val Dataset

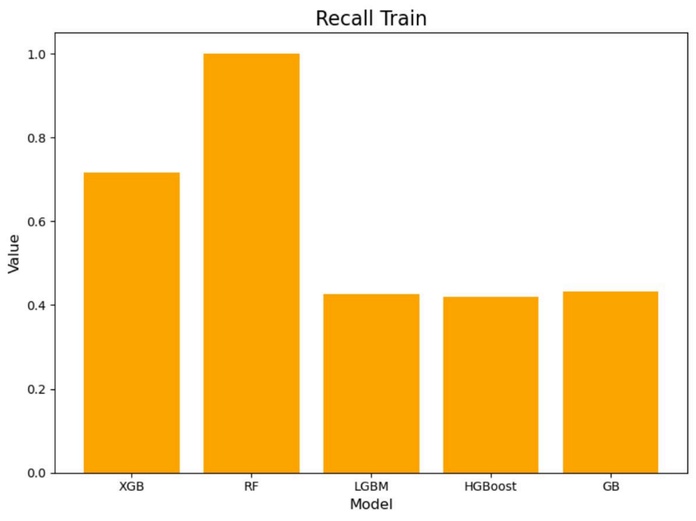




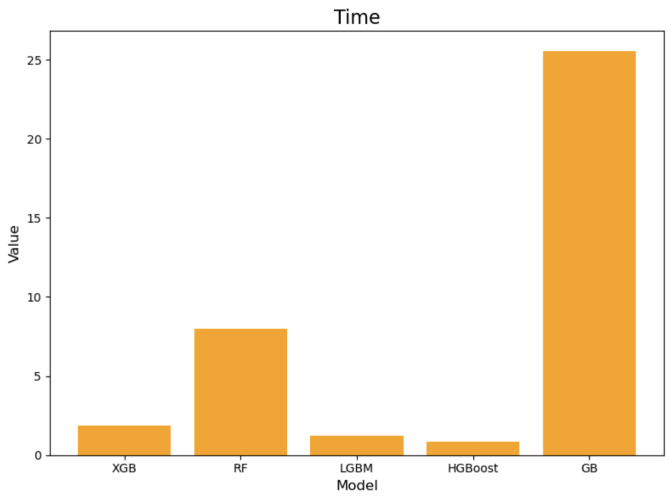
Cross-Validation Val Dataset



Cross-Validation Train Dataset



Cross-Validation Time


Time
Value
XGB
RF
LGBM
HGBoost
GB
Model