MACHINE LEARNING PROJECT

Master in Data Science and Advanced Analytics

NOVA Information Management School

Universidade Nova de Lisboa

Machine Learning Handout

To Grant or Not to Grant

Group 26

Pedro Santos, 2040295

Diogo Correia, 20211586

Oumaima Hfaieh, 20240699

Rita Morgadito, 20240611

Duarte Miguel, 20240608

Fall/Spring Semester 2024-2025

TABLE OF CONTENTS

Table of Contents

Abstract: 5

1. Introduction: 6

2. Data Exploration 7

3. Data Preprocessing 8

3.1. Incoherencies 8

3.2. Missing Values 9

3.2.1 BINNING 9

3.2.2 Reasoning Based Imputation 9

3.2.3 Mode 9

3.2.4 KNN Imputer 9

3.3 Scalers and Encoders 9

3.4 Feature Engineering 10

3.5. Feature Selection 10

3.5.1 Filter Methods 10

3.5.2 Wrapper Methods 11

3.5.3 Embedded Methods 11

4. Model Assessment: 11

4.1. Optimization: 12

4.1.1 Randomized Search 12

4.1.2 Grid Search 12

5. Open Ended 13

5.1 Agreement Reached Target Variable 13

5.2 Prediction Interface 14

5.2.1 Software Used 14

5.2.2 PROBLEM encountered 15

6. Conclusion 16

7. Appendices: 17

7.1. Methods not covered in class 17
7.2. Appendix A: Numerical Variables 17
7.3. Appendix B: Date Variables 19
7.4. Appendix C: CATEGORICAL Variables: 20
7.5. Appendix D: Preprocessing Options: 22
7.6. Appendix E: Feature Selection Graphs: 23
7.7. Appendix F: Modelling and Optimization Options: 26
7.8. Appendix G: WCB Web Interface 28
8. References 30

ABSTRACT:

This project aimed to build a predictive model to assist the New York Worker's Compensation Board (WCB) in regulating compensation, disability, and other critical factors related to workplace injuries. To support this effort, the WCB provided a dataset of over 550,000 resolved claims, ranging from 2020 to 2022, along with an additional dataset of approximately 350,000 unresolved claims from 2023 onwards.

The project followed a structured methodology, beginning with extensive data exploration and visualization to identify key patterns and insights. This was followed by rigorous data preprocessing to address inconsistencies, missing values, outliers, and variable transformations, including encoding and scaling. Feature engineering and selection played a pivotal role in preparing the dataset for the modeling stage, where multiple predictive models were implemented and rigorously evaluated. Optimization through hyperparameter tuning was essential to improve model generalization and performance. Finally, the results were analyzed across all datasets, and solutions were submitted to Kaggle for benchmarking.

Understanding the complexity of this multi-classification problem, with eight distinct classes, required careful selection of evaluation metrics. The F1 Macro score was chosen as the primary metric to optimize, balancing precision and recall across all classes. Among the models tested, the XGBoost emerged as the top performer, achieving a training score of approximately 0.48, a validation score of around 0.44, and a score of 0.46 on Kaggle (computed with 20% of the final data).

Keywords: Machine Learning, Data Exploration, Data Preprocessing, Feature Selection, CatBoost Classifier, Multi-Classification, F1 Macro, Kaggle, Workplace Claims

1. INTRODUCTION:

The emergence of increasingly advanced AI techniques and models is expected to reduce or significantly facilitate tedious and high-maintenance tasks in various fields. The growing ability of predictive models to tackle complex challenges is critical to creating a more efficient and productive society, allowing humans to focus on managerial and psychological tasks where algorithms currently might fall short.

In many industries, particularly the insurance sector, companies face difficult challenges such as fraud detection and claim validation, where machine learning is already playing a key role in improving their ability to combat fraudulent claims. Similarly, the idea of developing a predictive model to assist with work-related claims is both logical and necessary. These tasks are time-consuming, resource-intensive and prone to transparency issues, leading to higher costs and inefficiencies.

With this in mind, we were approached by the New York Worker's Compensation Board (WCB), which provided us with data on resolved claims and a separate dataset of unresolved claims. The goal was to successfully build a predictive model capable of accurately classifying eight possible outcomes, despite significant variation in class probability. The dataset contained a wide range of attributes: temporal information such as accident dates, forms completed by the individual and their employer and more, demographic details including age and salary, as well as industry specific classifications, and accident-related insights.

This brings us to our central research question: "Can a predictive model accurately classify workplace accident outcomes and provide a competitive alternative to manually reviewed processes?"

Our approach began with a thorough visualization of each feature in relation to the target variable, allowing us to understand the separate class probabilities for each feature. Data pre-processing was a critical step, addressing inconsistencies, incoherencies and missing values, followed by appropriate coding and scaling of variables. Feature engineering and feature selection played a key role in preparing the dataset for modelling, with intuition and domain reasoning also guiding our decisions.

In the modelling phase, we went through an extensive evaluation and optimization process, fine-tuning our best-performing baseline models within the constraints of our computational resources. In addition, we performed a separate analysis on a specific target variable to determine if an agreement (binary classification) was reached prior to the WCB's outcome. To finalize our work, we designed an interface for the WCB that integrates our best-performing models. This tool allows users to enter relevant claim details and receive a preliminary prediction of the likely outcome, providing valuable insight before proceeding with their claims.

2. DATA EXPLORATION

The initial stage of the process involved an exploration of the provided dataset, with the objective of identifying patterns, structures and potential issues, while also drawing preliminary insights. This process began with the resolution of data type constraints, particularly those associated with temporal features. This was achieved by converting the data from the DD-MM-YYYY format into years, thus facilitating the subsequent analysis. Additionally, the data was carefully classified into three distinct categories: numerical, categorical, and time related. It is worth noting that some features classified as "numerical," such as code-related attributes, were identified as categorical.

1) For Numerical Values: Histograms were constructed to gain insight into the distribution of the data. The histogram is a graphical representation that is used to analyze the distribution of data and identify the point at which most of the data is concentrated. Boxplots also were plotted, which gives us a graphical representation of the distribution of data too but is a method that is primordial in aiding to identify any outliers. Furthermore, feature versus target histograms were elaborated. The results provided insights into the ways in which distributions vary across target classes, allowing us to get a better sense of imbalance across features.

2) For Categorical Values: Bar charts were useful for visualizing categorical data. The tool provided a transparent representation of the data, facilitating the identification of discrepancies and guiding the initial interpretation of features. Here, we managed to observe the disparity in some values that were contained in each feature, gaining an initial view on how these can be handled later.

The following key insights were revealed:

- Most ages fall within the range of 20 to 80. However, there are instances where the value is "0," which is likely indicative of missing data, and outliers, such as values exceeding 85 years of age **(Appendix A: Figure A5).**

- The presence of outliers is notorious in most cases, as well as some specific features that show a high percentage of missing values (i.e. “Average Weekly Wage”, “C-3 Date”, “IME-4 Count”, **(Appendix A: Figure A6, A7)** and **(Appendix B: Figure B4).**

- Code Features: These categorical features, represented as numerical values, require careful imputation in order to preserve their categorical nature **(Appendix A: Figure A2, A3, A4).**

- It is evident that there are several redundant features. A few pairs of features, such as those englobing codes and descriptions, convey identical information but in different formats.

- Odd Distributions: The “Number of Dependents” feature shows a uniform distribution from 0 to 6, which appears atypical and warrants further precautions **(Appendix A: Figure A1).**

- Carrier Name: Exhibits high variability with many similar names representing the same carrier, necessitating specific preprocessing to consolidate similar values.

- Carrier Type: While there are five distinct categories, consolidating these residual values cold reduce noise and improve interpretability **(Appendix C: Figure C2).**

- Residual Categories: Many categorical features display typical values with minor residuals, which may need consolidation to improve model performance **(Appendix C: Figure C1, C3, C4, C5).**

These insights informed preprocessing decisions, such as binning irregular distributions, aggregating residual values, providing quality insights on a possible pipeline for our project.

3. DATA PREPROCESSING

A significant element of our data treatment process was the resolution of inconsistencies, the correction of incomplete or erroneous data, the identification and exclusion of anomalies in values, and the scaling and encoding of variables in preparation for the feature selection stage. To guarantee the robustness of the process, the holdout method was employed, in which the dataset was divided into training and validation sets from the start of the preprocessing stage. This approach prevents data leakage and ensures that any preprocessing applied to the training set is appropriately replicated on the validation and test sets, thus ensuring consistent model evaluation.

Caution was taken regarding the application of transformations, including scaling, encoding and the imputation of missing values. These transformations were fitted exclusively to the training data set and subsequently applied to the validation and test sets. This ensures that the validation and test sets remain true "holdout" sets, free from bias introduced by the training data set. Stratification was applied during the split to preserve class distributions due to the severe class imbalance. The split allocated 25% of the data to the validation set, with the remaining 75% used to train the model.

3.1. INCOHERENCIES

The initial step involved the elimination of rows that were entirely null, except for the "Claim Identifier" and "Assembly Date" columns. This decision assumed that such rows are the result of deficiencies in the WCB's information management system, either in the storage of claims that did not progress or in the reflection of simple data entry errors. Imputation was deemed inappropriate for these rows, as the process of filling missing values for approximately 20,000 cases would result in the generation of "artificial" data that lacks any resolved claim associations.

Furthermore, the 'Claim Identifier' feature was excluded from predictive modelling, as it serves solely as an identifier and therefore lacks any predictive value. Consequently, it was designated as the index for the purpose of facilitating data management. In order to guarantee the uniformity of data representation across all datasets, the "OIICS Nature of Injury Description" column, which was entirely null, was removed. Similarly, the variables "Agreement Reached" and "WCB Decision," which are only present in the training set and can only be determined after claim resolution, were excluded.

3.2. Missing Values

3.2.1 BINNING

Different features were imputed by a handful of techniques, focusing on removing noise and improving the quality of our data. The general idea focused on aggregating marginal values, agglomerating others, and allowing the model to do most of the work and capture patterns. In this manner, we felt that for most features, which demonstrated higher counts of null values, or higher disparity between values, we decided to bin most features, allowing the model to precisely capture this behavior (i.e. “First Hearing Date”, “IME-4 Count”, “Average Weekly Wage”, “C-2 Date”, “C-3 Date”).

3.2.2 Reasoning Based Imputation

An intuitive imputation approach was applied to date-related features (Birth Year, Accident Date, Age at Injury), which followed the simple equation:

Accident Date = Birth year + Age at Injury.

Evidently, the process is quite intuitive, this simple equation allows us to impute more accurate values, instead of using another type of imputer which could introduce more noise than actual precision.

3.2.3 Mode

The mode was applied uniquely to “Code” and “Description” pairs. For each pair, the most recurrent value was applied, these codes are representative of a category, hence, the mean or median or any other type of numerical imputer would be inappropriate. However, attempts to aggregate, with basis on external information, did not improve model performance, as scores slightly reduced.

3.2.4 KNN Imputer

As for KNN Imputer, we utilized it mostly as fallback for minimal values, with default parameters. KNN Imputer elaborates imputation for missing values based on the distances between K closest data points. It's important to note that we used default values (i.e. Euclidean Distance, and 5 neighbors), but by tuning these parameters we could obtain different results. As KNN relies on distance calculations, scaling was performed beforehand to ensure uniformity.

3.3 Scalers and Encoders

For scaling, several techniques were tested, leaving us to one final choice which ultimately also covers effectiveness in handling outliers. Unlike Standard or MinMax scalers, Robust Scaler scales features based on the IQR range, making it less sensitive to extreme values. Furthermore, by using this scaler, we tackle two problems at once, handling outliers as well as maintaining a coherent dataset. Nevertheless, refer to **(Appendix D: Table D1**) for a more comprehensive pick on Scalers and how these impacted our models.

As for Encoders, various techniques were explored, but our choices were mainly made based on how performance fluctuates. Therefore, the final options consisted of Frequency Encoder for our features and Label Encoder for our target variable. The decision to use Frequency Encoding for predictors was made because it converts categorical variables into numerical values, enabling feature selection methods and predictive models to process categorical data in a format that they can handle. A detailed analysis of the encoding techniques can be found in the following Table:

Encoders	Features	Purpose	Impact	Conclusion
Frequency Encoder	All	Encode values by frequency count.	Models showed promising results using this method.	Used
Frequency Encoder + One Hot	One Hot: "ADR", "Attorney/Representative", "Gender", "Medical Fee Region", "COVID-19 Indicator" Frequency: All Others	Obtain the best combination possible of encoding. By using one-hot on less sparse features.	Even though the approach seemed promising, most models did not react positively with this approach.	Not Used
Label Encoder	Target	Give a numerical class to our target variable.	Used in every implementation.	Used

Table D2: Encoder

3.4 Feature Engineering

Two new features were formulated, one being "*COVID_Industry*" where essentially the goal was to observe industries where COVID-19 may have dominated versus others that seemed less susceptible. The feature combined "*COVID-19 Indicator*" with "*Industry Code*", highlighting sectors that were deeply affected by the pandemic.

The high cardinality of "*Zip Code*", combined with domain reasoning, incited the creation of a binary feature. This indicator represents the presence or absence of a Zip Code, enabling the model to account for this information without being too influenced by the large discrepancy in unique values.

3.5. Feature Selection

Feature selection was essential in our project to maintain an efficient dataset by focusing on relevant features. This process improved model performance whilst reducing the risk of overfitting by removing noise. For these reasons, we immediately decided to discard all "Descriptions" from the model, since they demonstrate duplicate information which is already captured in the corresponding codes.

3.5.1 Filter Methods

We began by identifying univariate variables with zero variance in the training split, as features with no variability hold no predictive value. Next, we calculated the correlation matrix, focusing on true numerical features while excluding encoded categorical variables and "Code" features. Features with correlations exceeding a threshold of 0.7 were marked for potential removal to mitigate redundancy **(Appendix E: Figure E1).**

3.5.2 WRAPPER METHODS

Recursive Feature Elimination (RFE) was applied using Logistic Regression and Random Forest Classifier. Both RFE's indicated that 22 would be the optimal number of features. Additionally, visualizations of F1 Macro score fluctuations were plotted to further support our findings (**Appendix E: Table E2**) and **(Appendix E: Table E3**).

3.5.3 EMBEDDED METHODS

In this section, we used Lasso Regression to measure feature importance. Lasso inherently tracks performance by shrinking less important feature coefficients to zero. It does so by adding regularization, which is described as a penalty to the loss function.

The selected features were based on majority votes, where features were discarded if at least 2 out of the 4 methods indicated to do so. Final selection can be observed in **(Appendix E: Table E4**) where "Zip Code Binary", "Alternative Dispute Resolution", "Number of Dependents" and "Assembly Date" were dropped.

4. MODEL ASSESSMENT:

A general evaluation function was defined, in order to determine the best performing baseline models, optimized to compute F1-Macro Scores for training and validation sets. Furthermore, this was a key step in assessing the generalizability of the models, a key metric that we took many precautions on, ensuring reliable performance on unseen data, notably Kaggle submissions, which served as benchmarking.

This function provided great utility, since the baselines were a great insight on how performance oscillates, allowing us to iteratively test different preprocessing techniques, and optimize results. Hence, six different models were tested: Logistic Regression, Decision Trees, Random Forest Classifier, Gradient Boosting Classifier, XGBoost and finally CatBoost. The results varied as well as some model's overfitting severely to the train data, as can be seen in below in **Table F1**, where notably XGBoost and CatBoost showed promising results, balancing performance with overfitting.

Finally, one of the main objectives of this work was to implement a Neural Network. Multiple experiments were conducted, focusing mainly on defining an optimal architectural structure of hidden layers and neurons. Ultimately, despite defining an optimal structure, the model did not end up being the best performer. Choice of scalers, encoders and ultimately the whole preprocessing pipeline could be a key motive as to why the model did not end up performing as expected.

Given these findings, we proceeded with several optimization techniques, and due to time and computational resources available, the three best performing models were chosen to conduct further analysis: Cat Boost, XGBoost and MLP Classifier.

Models	F1-Macro Score	Hyper Parameterization	Goodness of Fit	Conclusion
Logistic Regression	Train: 0.314 Validation: 0.314	Not Performed - Reason: Low Performer	No Apparent Overfitting: Good Generalizability	Not Used - Low Performance & Predictability of the Minority Class
Random Forest	Train: 0.999 Validation: 0.401	Not Performed - Reason: Time Efficiency & High Overfitting	Clear Overfitting	Not Used
Decision Tree	Train: 0.999 Validation: 0.384	Not Performed - Reason: Time Efficiency & High Overfitting	Performance and Overfitting: Reasonable after Tuning	Better than Random Forest - Not Used
Gradient Boosting	Train: 0.407 Validation: 0.401	Not Performed - Reason: Low Performer	Good Generalizability	Bad Performance - Not Used
XGBoost	Train: 0.623 Validation: 0.443	F1 Macro Train: 0.483 F1 Macro Val: 0.4372	Some Minor Overfitting (0.04 p.p)	Best Performer - Used
CatBoost	Train: 0.569 Validation: 0.429	F1 Macro Train: 0.467 F1 Macro Val: 0.422	Some Minor Overfitting (0.04 p.p)	Good Performer - Not Used
Neural Network	Train: 0.433 Validation: 0.417	F1 Macro Train: 0.43 F1 Macro Val: 0.425	N/A	No further tuning was elaborated - Due to Time Efficiency

Table F1: Modelling Summary

4.1. OPTIMIZATION:

To avoid potential leakage caused by repeated data splits in default Randomized and Grid Search implementations, we utilized a Predefined Split strategy, where both training and validation sets were combined for optimization.

4.1.1 RANDOMIZED SEARCH

Randomized Search was our initial approach for hyperparameter optimization, leveraging its efficiency to explore a larger parameter space. It samples parameter combinations randomly and evaluates them across a defined number of iterations. This allowed us to identify promising parameter ranges for further fine-tuning.

4.1.2 GRID SEARCH

Following the previous insights, we adapted the grid, by reducing considerably the combinations to a manageable size. This was elaborated, due to the computational intensity required by Grid Search, which exhaustively tests all combinations. The objective was not simply to maximize validation set

performance, but to ensure both sets had similar performances, minimizing the risk of overfitting, providing security in generalizability.

For this purpose, here are the observations across our 3 best performing models:

- Neural Network: Randomized Search yielded satisfying results, yet further optimization was avoided due to training time and underperformance in comparison to other models, likely due to preprocessing constraints.

- CatBoost: Although in past experiments, this model's performance was very strong, some signs of overfitting persisted, and performance was still lacking in comparison to XGBoost.

- XGBoost: The model emerged as our top performer, with controlled overfitting, we still managed to achieve very competitive results. It achieved a benchmark score of 0.46, outperforming CatBoost (0.45) and the Neural Network. Additionally, further manual tuning of parameters (mainly through controlling *"maximum_depth"*) helped us reduce the gap between training and validation performance within the 0.05 percentage point threshold.

We successfully reduced overfitting to within the defined threshold, ensuring generalizability across training and validation sets. The results for the three models, along with their respective Kaggle scores, can be seen below:

Models	Randomized Search F1 Macro Scores	Grid Search F1 Macro Scores	Manual Tuning F1 Macro Scores	Overfitting	Kaggle Score
CatBoost	Train: 0.622 Validation: 0.426	N/A	Train: 0.467 Validation: 0.422	0.045 Difference	0.455
XGBoost	Train: 0.563 Validation: 0.443	Train: 0.562 Validation: 0.438	Train: 0.483 Validation: 0.437	0.046 Difference	0.463
MLP Classifier	Train: 0.43 Validation: 0.425	N/A	N/A	Very Residual	N/A

Table F2: Hyper parameterization and Benchmarking

5. Open Ended

5.1 Agreement Reached Target Variable

As part of our implementation, we also decided to elaborate a model which plans to predict the "Agreement Reached" binary variable, indicating whether an agreement was reached prior to a WCB Decision or not. The process was identical to the main notebook in terms of data preparation and

preprocessing. However, feature selection varied slightly, removing a different set of variables. For instance, “Alternative Dispute Resolution” was maintained whereas “Birth Year” and “COVID-19 Indicator” were removed based on the same procedures outlined in section 3.5.

Given the severe class imbalance in the target variable, with over 90% of instances belonging to the majority class “0”, we opted for the F1 Macro score, while also giving special attention to the f1 standard score for the minority class. For this research, CatBoost emerged as the top performer achieving a macro score on the validation set of around 0.7, and a standard f1 score of 0.45 on the minority class, which we found to be a decent performance. Additionally, considering that this task does not reflect the same implications as a medical diagnosis for example, such severity in criteria is not expected. Nevertheless, we could expect to boost performance by applying techniques that could artificially create samples of the minority class (e.g., SMOTE), allowing the model to better capture the minority class’s details. In light of the above, a summary of the results can be found in **(Appendix F: Table F3**).

5.2 Prediction Interface

Although the models achieve relatively satisfying scores, for both target features, we believe an implementation presents significant values for the WCB. By integrating two new tabs into the WCB’s processes, the system allows for users to get their own predictions effectively. This provides users with better insights into possible compensation or agreement outcomes based on actual data, reducing the number of unnecessary processes submitted to the WCB and improving operational efficiency.

5.2.1 Software Used

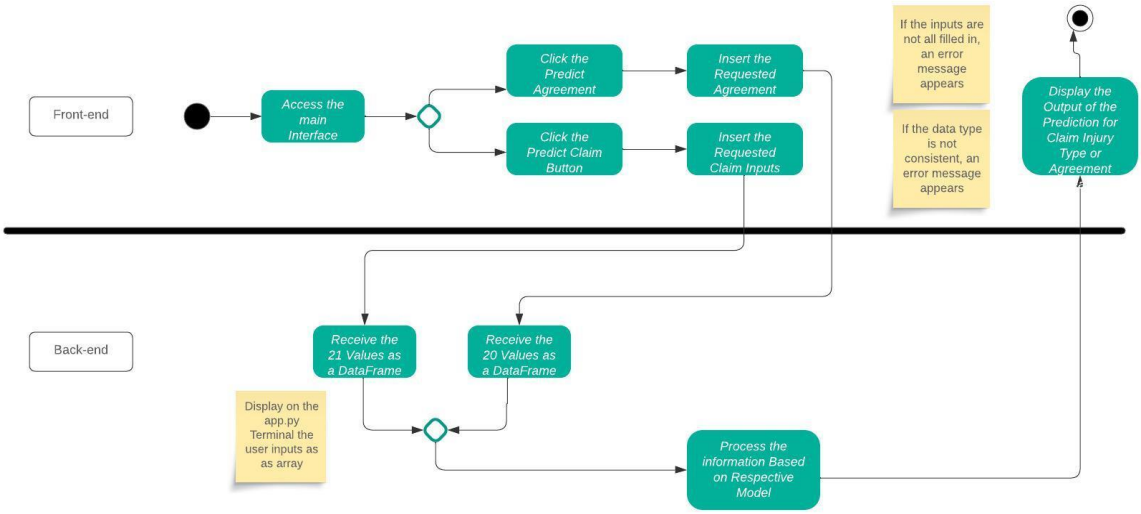


Diagram G4: Diagram of Activity - Interface

- Front-end - The development of the front-end used the essence of web programming, based on the creation of interfaces in HTML, CSS and JavaScript. The first interface replicates the WCB website homepage, with two new buttons linking to the forecast tools **(Appendix G: Figure G1)**. Both these pages allow users to input data into a text field and generate predictions

at the click of a button **(Appendix G: Figure G2) and (Appendix G: Figure G3)**. It's also important to note that in the top right-hand corner of the forecasting interfaces, there is a help link describing what the required fields represent, to assist with data entry.

- Back-end - The back-end was developed using Flask, a Python framework, used to handle all our data processing. Connecting the front-end inputs to our predictive models whilst returning the outputs to the user. All the data coming from the front end can be seen in the *app.py* terminal, as well as the forecast. Both the mapping of the features and the *label mapping* are carried out in this file, as well as the loading of the models used.

- Model Integration (.pkl and .cdm files) - Two separate models were deployed, for our "Claim Injury Type" target, an XGBoost with 21 features was stored as a .pkl file using the Joblib library in Python. Whereas for "Agreement Reached", the model CatBoost consisting of 20 final features was extracted via .cdm. Both models are stored in the *"model"* folder.

5.2.2 PROBLEM ENCOUNTERED

Although forecasts can be generated using the selected models, a key challenge was the preprocessing pipeline. The models were scaled and encoded during development to improve effectiveness and as a result, the model is trained on purely numerical data. Hence, users can only enter numerical data as their inputs for predictions. Evidently, this is problematic when features like "*Gender*", need to be filled (i.e., "M" or "F"). This complexity increased for dates and categorical variables with great disparity.

To solve this, we extracted both the scaler and encoder objects into a *.pkl* file and attempted to "*.inverse_transform*" the modifications underwent. However, numerous errors persisted, which undertook a severe debugging process, nevertheless, the following error remained: "*This ColumnTransformer instance is not fitted yet. Call 'fit' with appropriate arguments before using this estimator.*". Despite exploring alternatives and including the pipeline suggested on StackOverflow, we were unable to resolve this issue.

Finally, we firmly believe that this problem was not too far away from our reach and that with further refinement in future iterations, users will be able to input real values directly (e.g., text and date types) on the WCB's website. Continuous efforts will be made in order to provide a successful implementation for our users.

6. CONCLUSION

Having developed an extensive and detailed process to train a model to predict WCB decisions on workers' compensation claims, we revisit our central question: Does this immediately make this technology a direct replacement for human intervention in this area? Understandably, perfection was not achieved, but in machine learning, it is often indicative of overfitting to our trainining set. Nevertheless, the results were satisfactory and in line with our expectations, given the major challenge posed by the high imbalance of the dataset.

The results suggest that this approach is very competitive versus manually reviewing each claim, mainly due to the time inefficiency of the manual process. However, we question whether the goal should be to create a system that completely outperforms human capabilities or one that serves as an invaluable tool to augment human efforts. We believe that the latter is indeed the optimal way forward. In our implementation of the interface for the WCB, users can receive preliminary predictions for claims, through the model that was elaborated, thereby reducing the number of actual claims submitted and reducing the workload for manual review.

Despite these promising results, several limitations in our approach could be addressed to further improve performance. For example, cross-validation was not performed, which could have helped to mitigate overfitting and ensure model stability across different data folds. In addition, oversampling of minority classes by generating synthetic data points could be explored to improve model predictability for minority classes.

Additionally, due to computational and time constraints, we were not able to perform exhaustive fine-tuning to achieve the best hyperparameters. With more extensive tuning, it is likely that further improvements could be achieved. Finally, while we considered using a neural network approach, some of the pre-processing adjustments required for these models were challenges that we couldn't resolve in order to achieve optimal results. Though beyond the scope of this project, a deep learning-based solution at a later stage could further position the model as a strong and competitive tool in the claim’s assessment process.

This work demonstrates the potential of machine learning to transform workplace claims management, offering a scalable and efficient alternative to manual processes. While it has limitations, the results highlight significant progress, with room for further refinement. With continued innovation, this approach can set a new standard for handling claims, balancing human expertise with the power of advanced predictive models, which in some sectors such as insurance, are already being implemented.

7. APPENDICES:

7.1. METHODS NOT COVERED IN CLASS

- **Frequency Encoder:** Replaces our categorical values with their frequency of occurrences in the dataset. This method provides a simple numerical representation of categories whilst maintaining some information about their distribution.

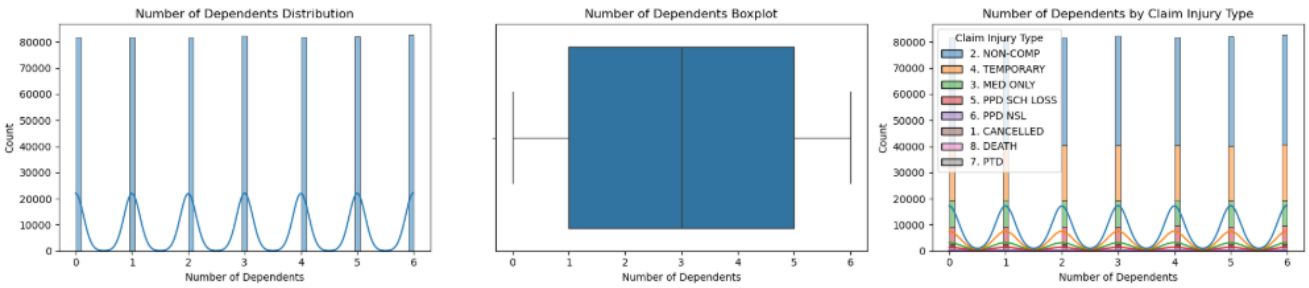
- **XGBoost:** XGBoost constructs an ensemble of decision trees, where each tree is trained to correct errors of the previous one. It's an advanced implementation of gradient boosting algorithms, designed for speed and performance.

- **CatBoost:** CatBoost is also an ensemble of decision trees, designed specifically for categorical data optimization. It handles categorical features natively without needing to extensively preprocess them. Excels in tasks dealing with imbalanced and noisy data sets.

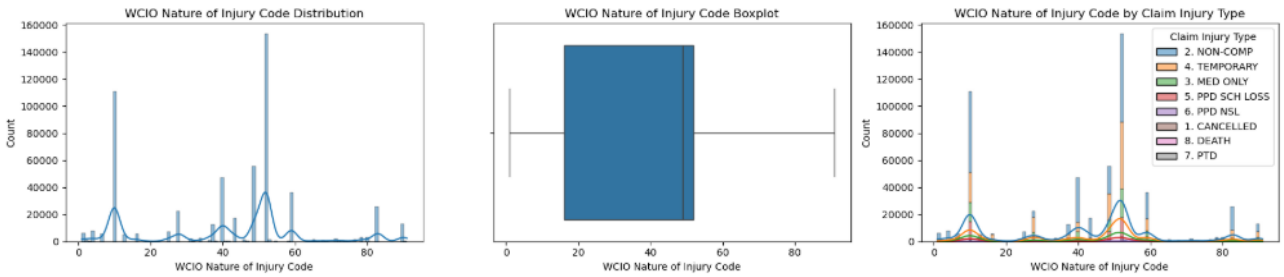
- **One Hot Encoder:** Transforms categorical data into binary vectors, where each value is represented by a distinct binary outcome (0/1). Zero standing for absence of the value in said column where 1 indicates the contrary.

7.2. APPENDIX A: NUMERICAL VARIABLES

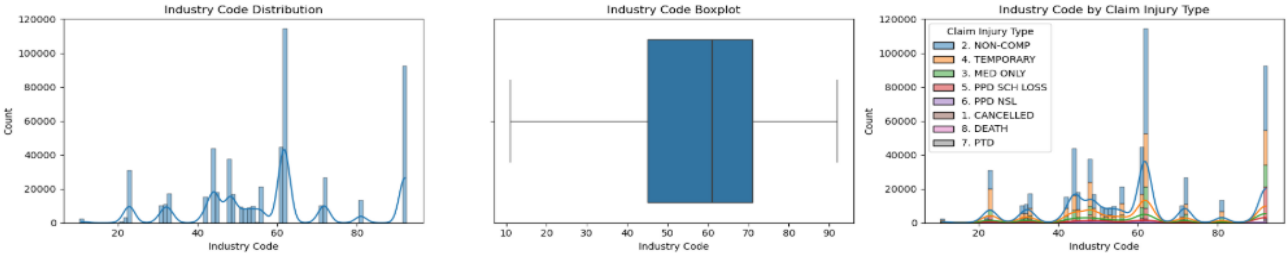
7.2.1. FIGURE A1: NUMBER OF DEPENDENTS



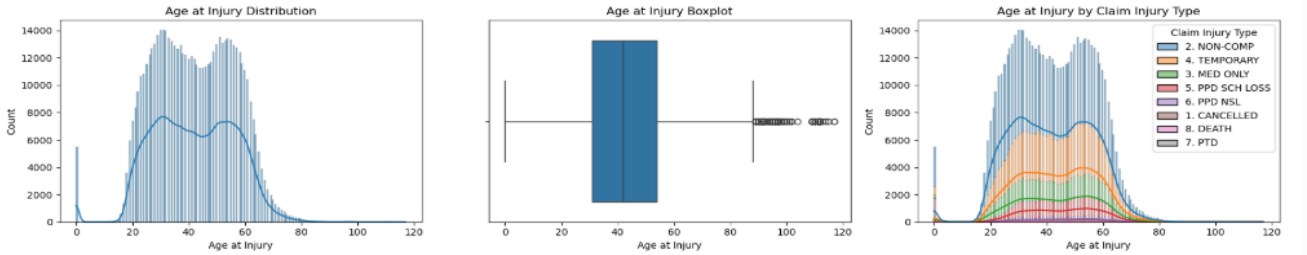
7.2.2. FIGURE A3: WCIO CAUSE OF INJURY CODE



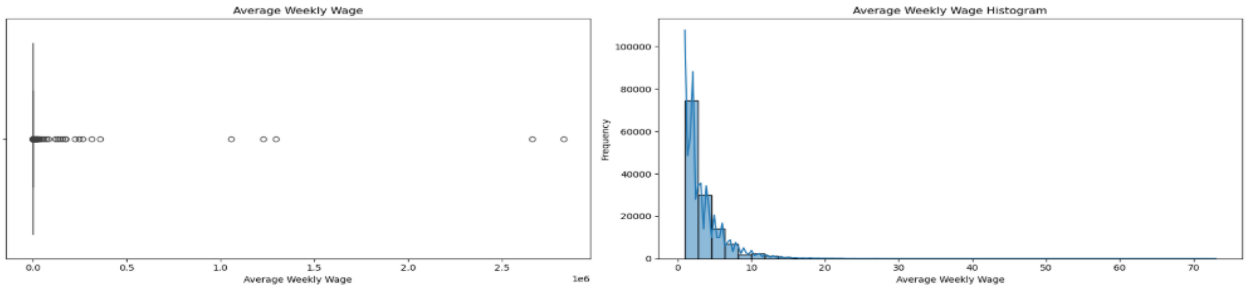
7.2.3. Figure A4: Industry Code



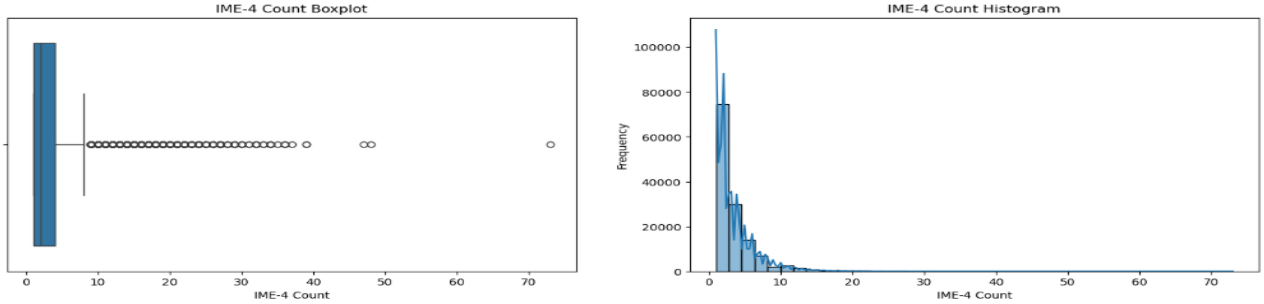
7.2.4. Figure A5: Age at Injury:



7.2.5. Figure A6: Average Weekly Wage

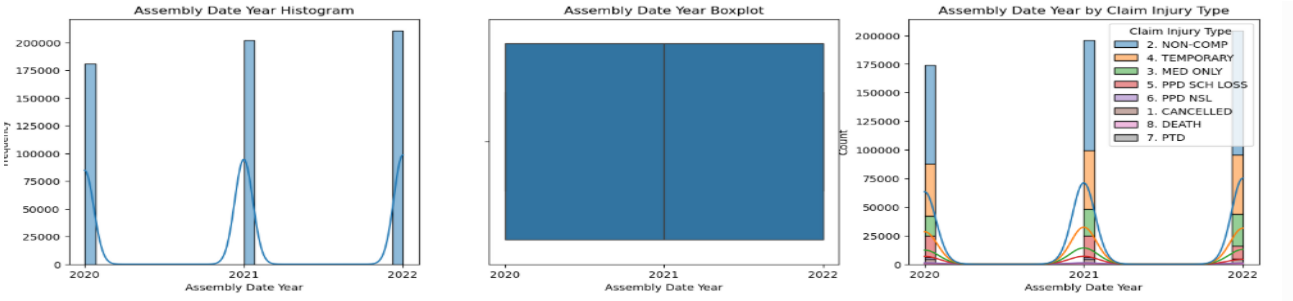


7.2.6. Figure A7: IME-4 Count

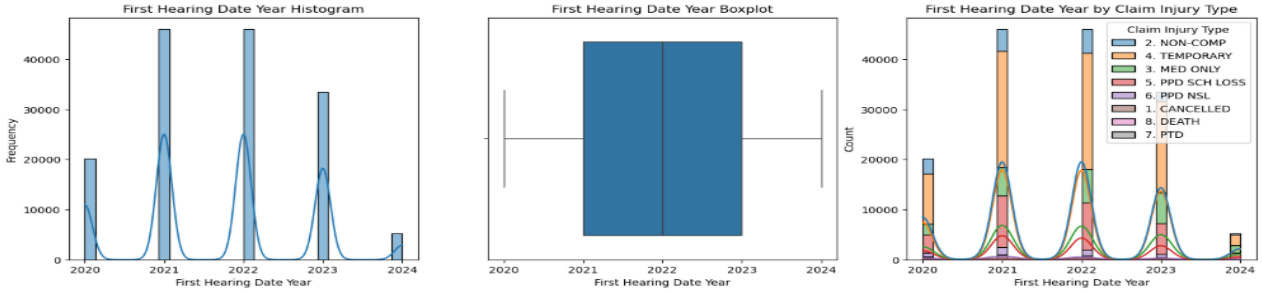


7.3. Appendix B: Date Variables

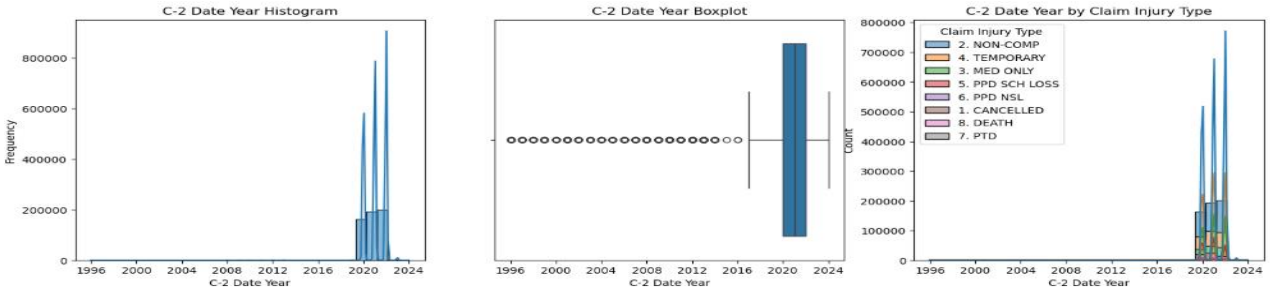
7.3.1. Figure B1: Assembly Date



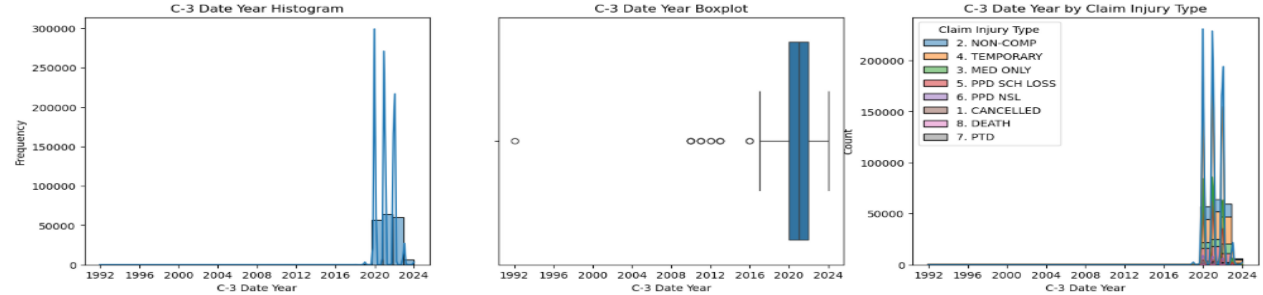
7.3.2. Figure B2: First Hearing Date



7.3.3. Figure B3: C-2 Date

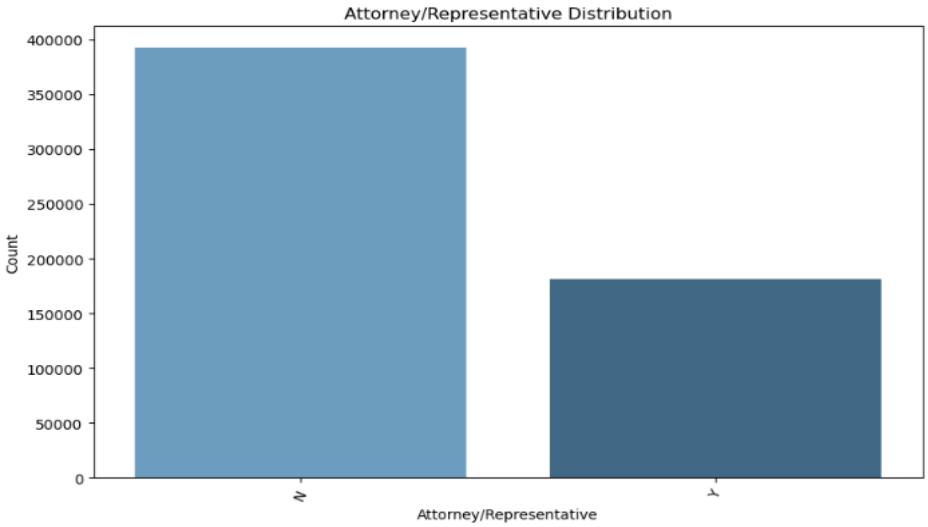


7.3.4. Figure B4: C-3 Date

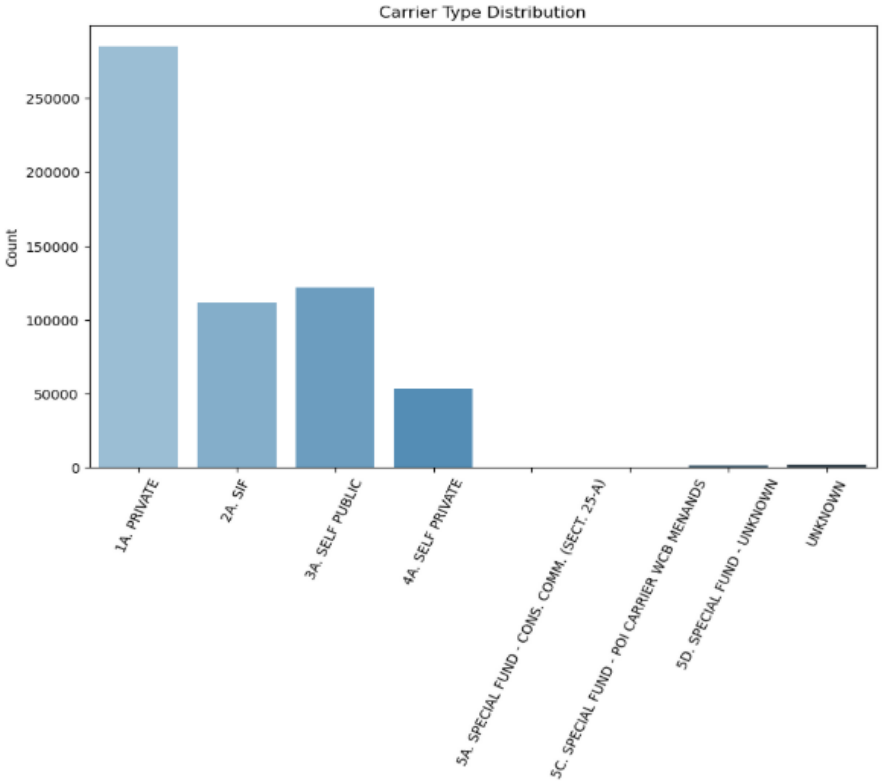


7.4. APPENDIX C: CATEGORICAL VARIABLES:

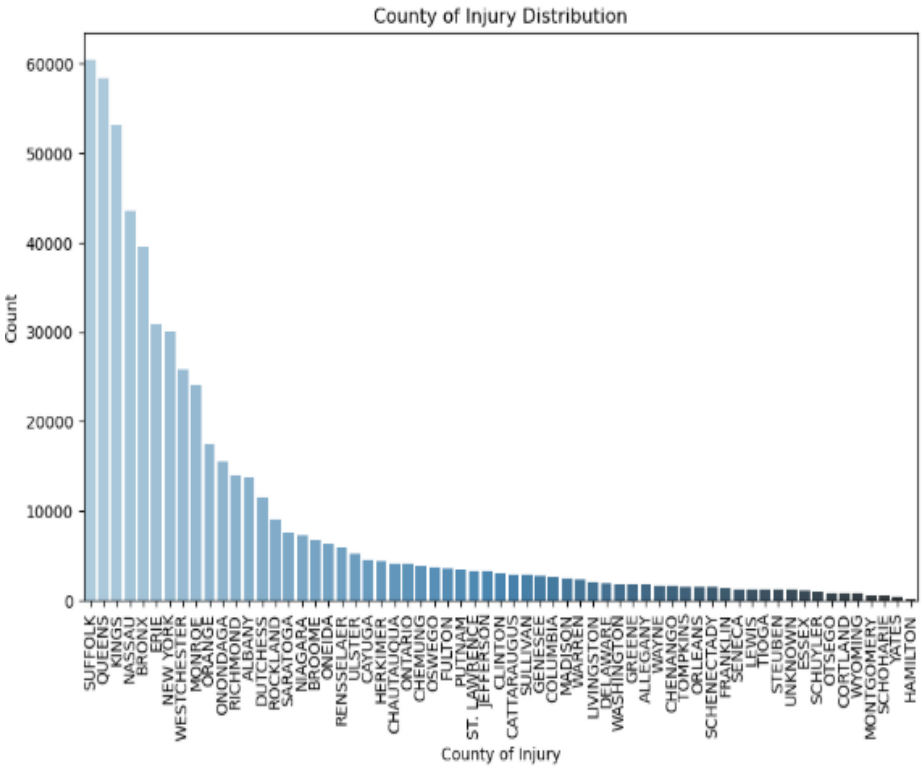
7.4.1. FIGURE C1: ALTERNATIVE DISPUTE RESOLUTION



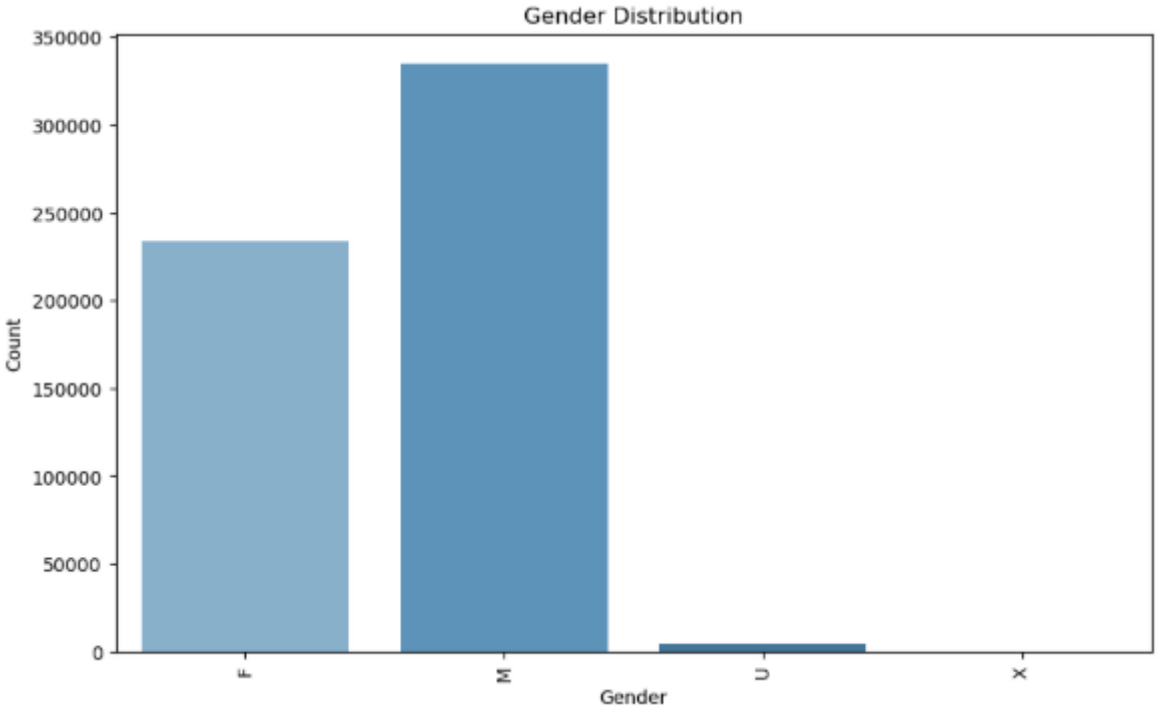
7.4.2. FIGURE C2: CARRIER TYPE



7.4.3. FIGURE C3: COUNTY OF INJURY



7.4.4. FIGURE C4: GENDER



7.5. Appendix D: Preprocessing Options:

7.5.1. Figure D1: Scalers

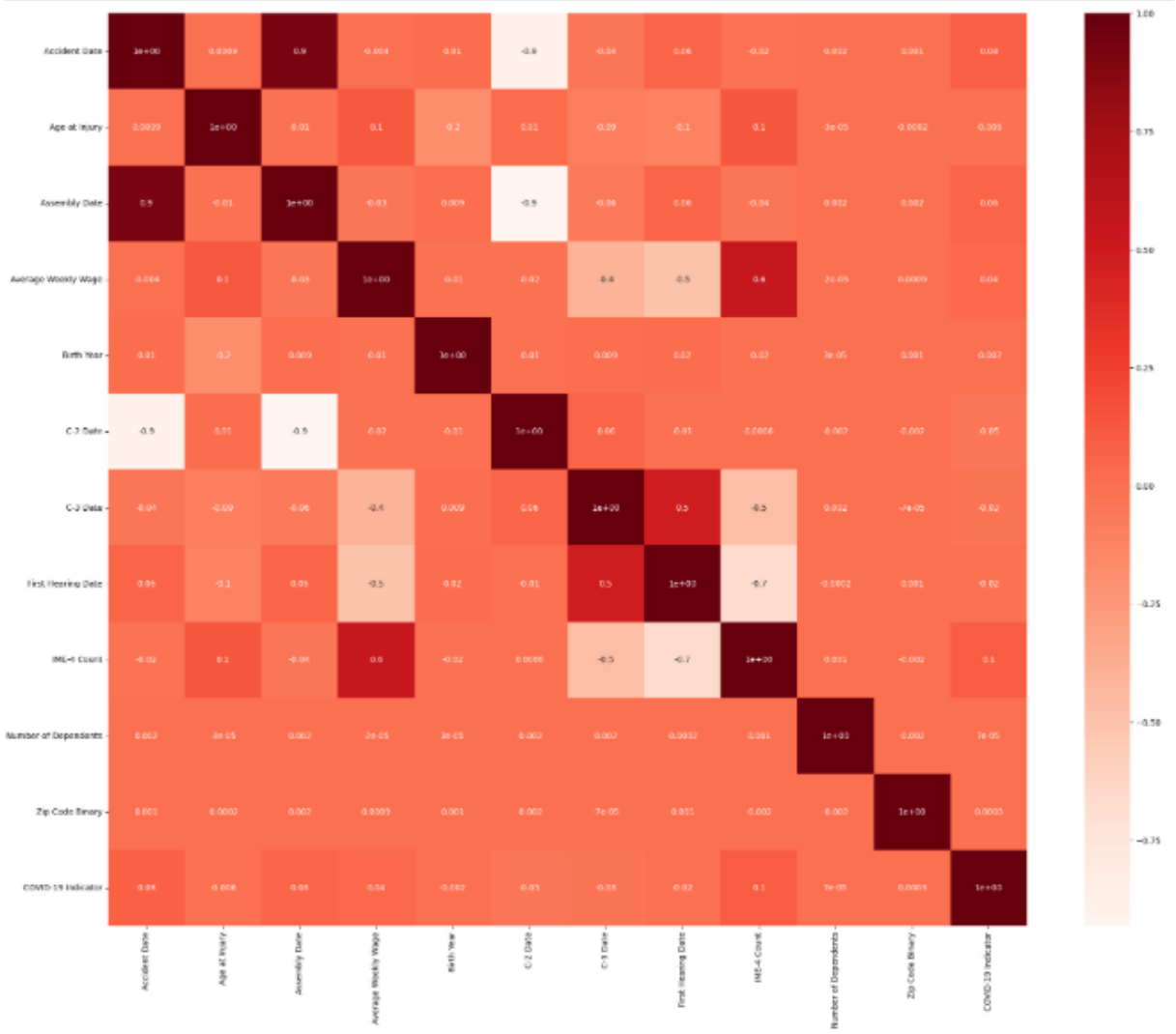
Scalers	Configurations	Purpose	Impact	Conclusion
Robust Scaler	Default	Scaled our features based on median and IQR range. Helped in taking care of outliers and extreme values.	Optimal results on final models	Used
MinMax Scaler	Feature Range: (-1,1)	This option was explored given our mission to implement a Neural Network.	Better Performance on NN.	Unused, given that NN was still outperforming.

7.5.2. Figure D2: Encoders

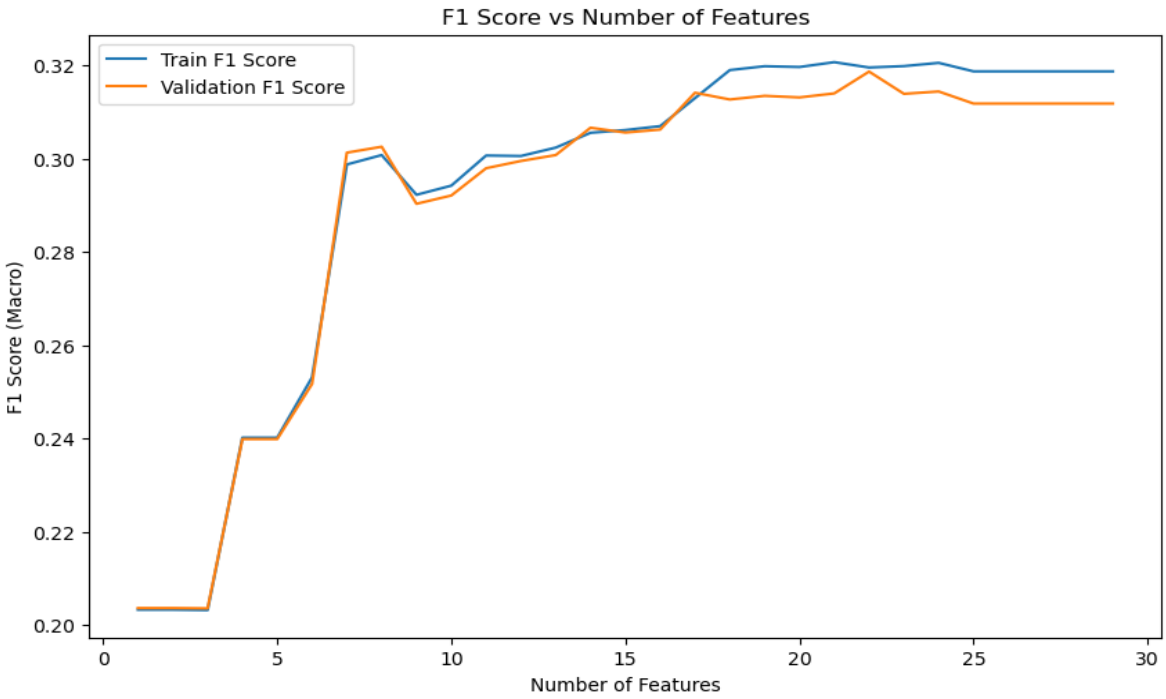
Encoders	Features	Purpose	Impact	Conclusion
Frequency Encoder	All	Encode values by frequency count.	Models showed promising results using this method.	Used
Frequency Encoder + One Hot	One Hot: "ADR", "Attorney/Representative", "Gender", "Medical Fee Region", "COVID-19 Indicator" Frequency: All Others	Obtain the best combination possible of encoding. By using one-hot on less sparse features.	Even though the approach seemed promising, most models did not react positively with this approach.	Not Used
Label Encoder	Target	Give a numerical class to our target variable.	Used in every implementation.	Used

7.6. Appendix E: Feature Selection Graphs:

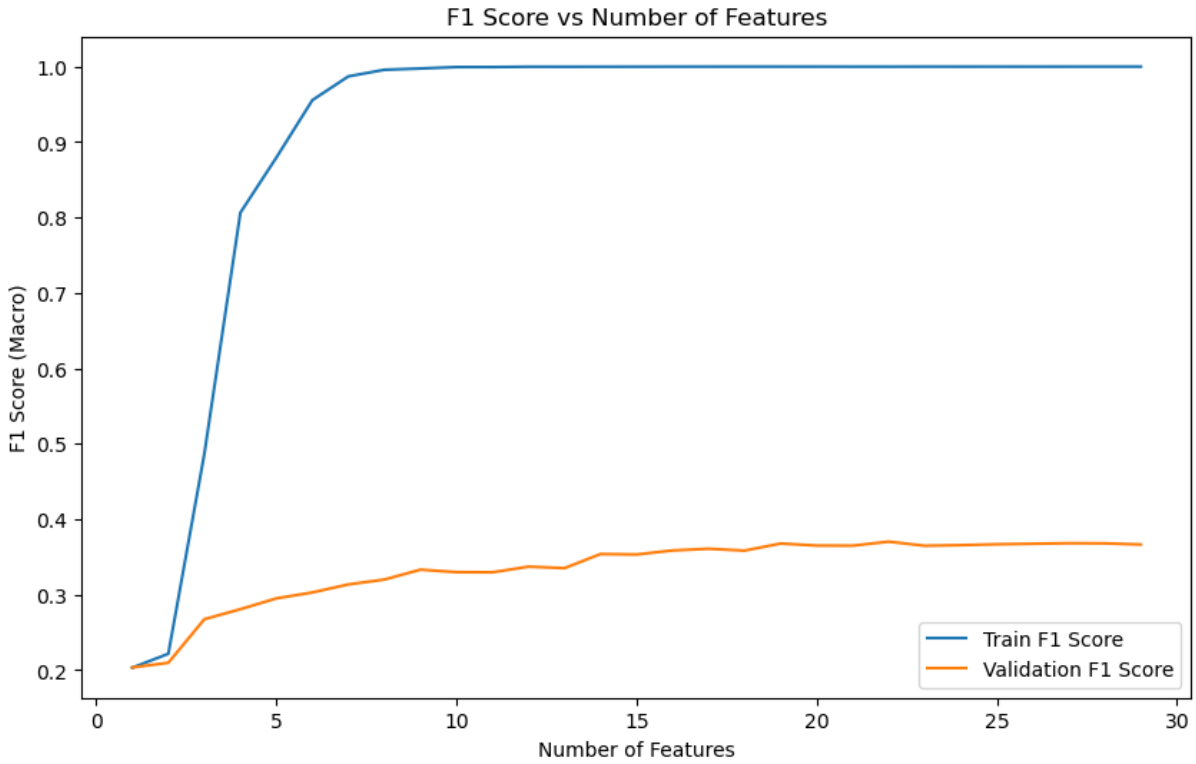
7.6.1. Figure E1: Correlation Matrix



7.6.2. FIGURE E2: F1 SCORE IN RFE USING LOGISTIC REGRESSION



7.6.3. FIGURE E3: F1 SCORE IN RFE USING RANDOM FORREST



7.6.4. TABLE E4: FEATURE SELECTION SUMMARY AND DECISION

Variables	Spearman	RFE LR	RFE RFC	Lasso	Conclusion
COVID_industry	Keep	Keep	Keep	Keep	Include
Number of Dependents	Keep	Remove	Keep	Remove	Discard
Zip Code Binary	Keep	Keep	Remove	Remove	Discard
Birth Year	Keep	Remove	Keep	Keep	Include
Alternative Dispute Resolution	Keep	Keep	Remove	Remove	Discard
County of Injury	Keep	Remove	Keep	Keep	Include
Assembly Date	Correlated	Keep	Keep	Keep	Discard
Accident Date	Correlated	Keep	Keep	Keep	Include
COVID-19 Indicator	Keep	Keep	Remove	Keep	Include

7.7. APPENDIX F: MODELLING AND OPTIMIZATION OPTIONS:

7.7.1. TABLE F1: MODELLING SUMMARY

Models	F1-Macro Score	Hyper Parameterization	Goodness of Fit	Conclusion
Logistic Regression	Train: 0.314 Validation: 0.314	Not Performed - Reason: Low Performer	No Apparent Overfitting: Good Generalizability	Not Used - Low Performance & Predictability of the Minority Class
Random Forest	Train: 0.999 Validation: 0.401	Not Performed - Reason: Time Efficiency & High Overfitting	Clear Overfitting	Not Used
Decision Tree	Train: 0.999 Validation: 0.384	Not Performed - Reason: Time Efficiency & High Overfitting	Performance and Overfitting: Reasonable after Tuning	Better than Random Forest - Not Used
Gradient Boosting	Train: 0.407 Validation: 0.401	Not Performed - Reason: Low Performer	Good Generalizability	Bad Performance - Not Used
XGBoost	Train: 0.623 Validation: 0.443	F1 Macro Train: 0.483 F1 Macro Val: 0.4372	Minor Overfitting (0.04 p.p)	Best Performer - Used
CatBoost	Train: 0.569 Validation: 0.429	F1 Macro Train: 0.467 F1 Macro Val: 0.422	Minor Overfitting (0.04 p.p)	Good Performer - Not Used
Neural Network	Train: 0.433 Validation: 0.417	F1 Macro Train: 0.43 F1 Macro Val: 0.425	N/A	No further tuning was elaborated - Due to Time Efficiency

7.7.2. TABLE F2: HYPER PARAMETERIZATION AND BENCHMARKING

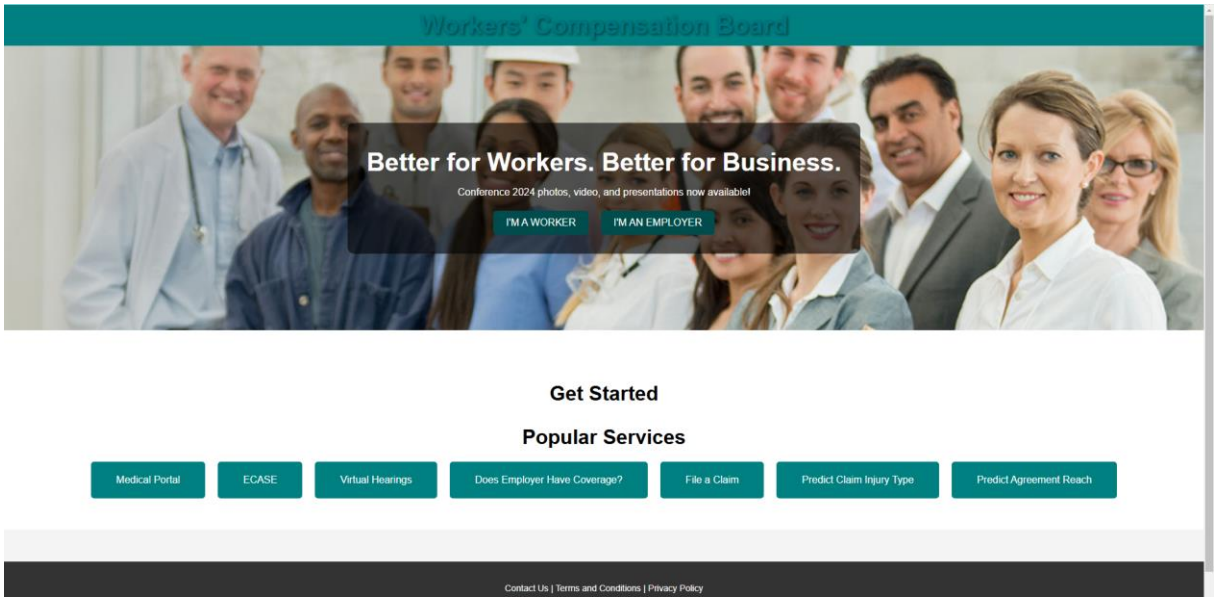
Models	Randomized Search F1 Macro Scores	Grid Search F1 Macro Scores	Manual Tuning F1 Macro Scores	Overfitting	Kaggle Score
CatBoost	Train: 0.622 Validation: 0.426	N/A	Train: 0.467 Validation: 0.422	0.045 Difference	0.455
XGBoost	Train: 0.563 Validation: 0.443	Train: 0.562 Validation: 0.438	Train: 0.483 Validation: 0.437	0.046 Difference	0.463
MLP Classifier	Train: 0.43 Validation: 0.425	N/A	N/A	Very Residual	N/A

7.7.3. TABLE F3: MODELLING SUMMARY – AGREEMENT REACHED

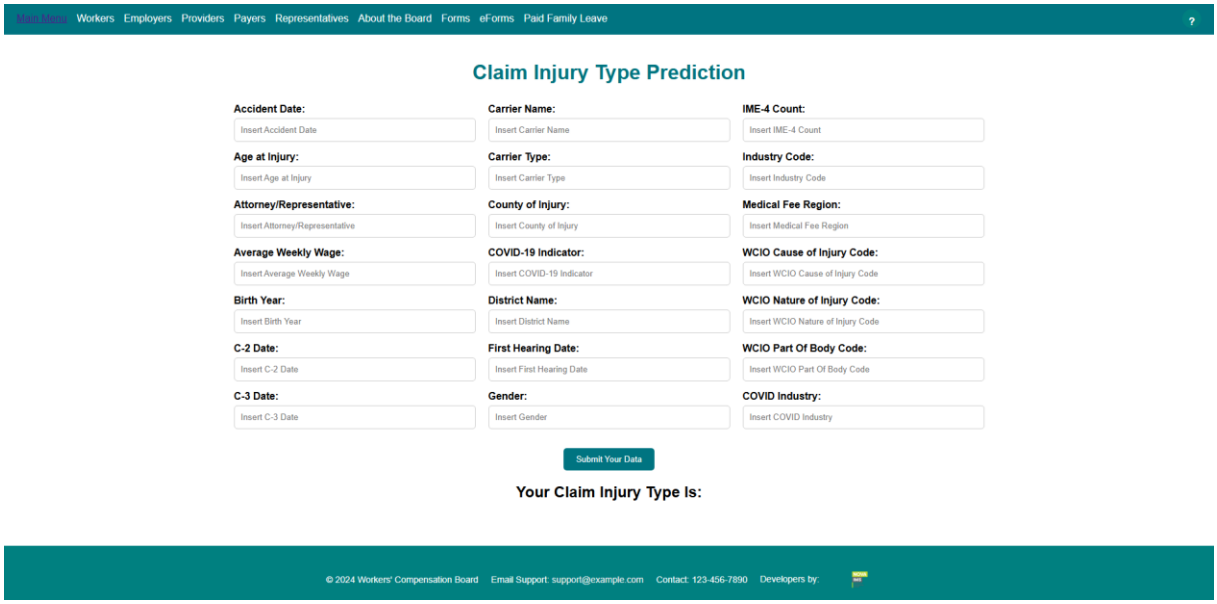
Models	F1-Macro Score	Standard F1-Score	Hyper Parameterization	Goodness of Fit	Conclusion
Logistic Regression	Train: 0.505 Validation: 0.505	Train: 0.035 Validation: 0.035	Not Performed	Good	Low Performance & Predictability of the Minority Class
Random Forest	Train: 0.989 Validation: 0.607	Train: 0.2385 Validation: 0.2385	Not Performed Reason: Time Efficiency	Clear Overfitting	Not Used
Decision Tree	Train: 0.999 Validation: 0.636	Train: 0.316 Validation: 0.316	Performed F1 Macro: 0.193	Performance and Overfitting: Reasonable after Tuning	Better than Random Forest, Not Used
Gradient Boosting	Train: 0.559 Validation: 0.559	Train: 0.141 Validation: 0.141	Not Performed	Good Generalizability	Bad Performance, Not Used
XGBoost	Train: 0.65 Validation: 0.62	Train: 0.263 Validation: 0.263	Performed F1 Macro: 0.237	Minor Overfitting (0.5 p.p)	Good Performer, Not Used
CatBoost	Train: 0.678 Validation: 0.624	Train: 0.270 Validation: 0.270	Performed F1 Macro: 0.7025	Good Generalizability After Control	Best Performer, Used

7.8. Appendix G: WCB Web Interface

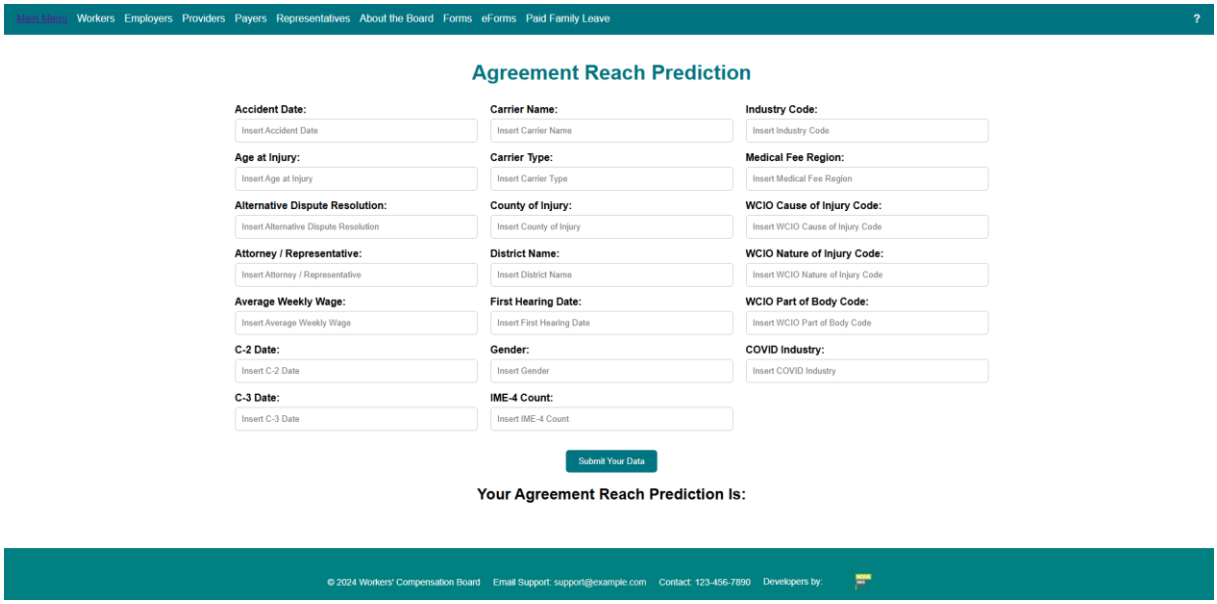
7.8.1. Figure G1: WCB Main Page



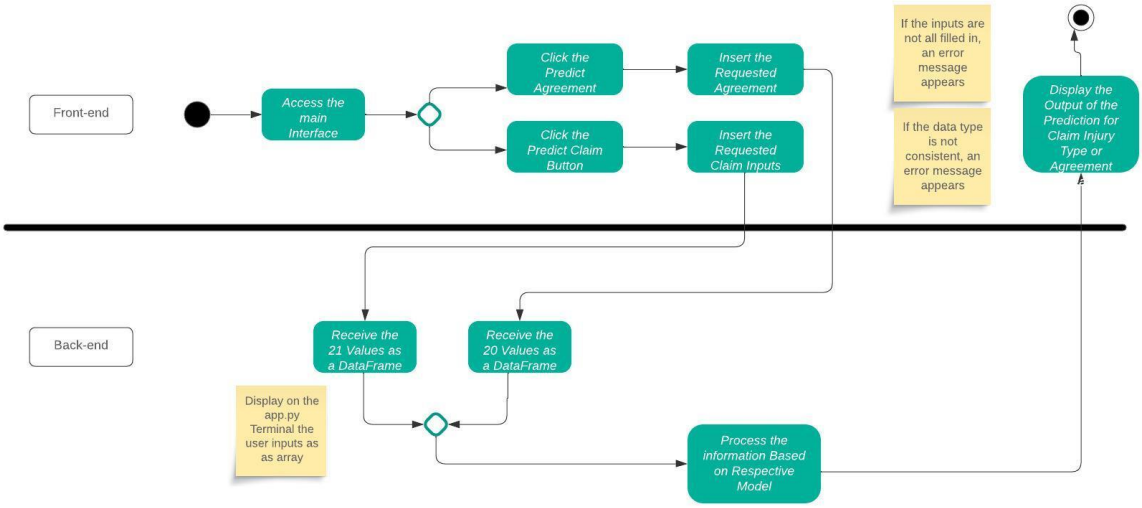
7.8.2. Figure G2: WCB Predict Claim Injury Type Page



7.8.3. FIGURE G3: WCB PREDICT AGREEMENT REACH PAGE



7.8.4. FIGURE G4: DIAGRAM OF ACTIVITY - INTERFACE



8. REFERENCES

1. Website Links

https://stackoverflow.com/questions/56556148/why-columntransformer-does-not-call-fit-on-its-transformers

https://stackoverflow.com/questions/69163930/python-valueerror-the-number-of-columns-in-this%20dataset-is-different-from-the-o

2. Github Repository

https://github.com/Pedro-P-Santos/Machine-Learning---Claim-Predictions-NOVA-IMS-

3. API or Documentation References

- XGBoost Developers. "XGBoost Documentation." Accessed [November, 2024]. *URL:* https://xgboost.readthedocs.io/en/stable/

- CatBoost Developers. "CatBoost: Gradient Boosting with Categorical Features Support." Accessed [November, 2024]. *URL:* https://catboost.ai/

- Scikit-learn Developers. "MLPClassifier: Neural Network Classifier (v1.5)." Accessed [November, 2024]. *URL:* https://scikit-learn.org/1.5/api/sklearn.neural_network.html

- Feature Engine Documentation. "CountFrequencyEncoder: User Guide." Accessed [November, 2024]. *URL:* https://feature-engine.trainindata.com/en/1.8.x/user_guide/encoding/CountFrequencyEncoder.html

- Scikit-learn Developers. "OneHotEncoder: API Documentation (v1.5)." Accessed [November, 2024]. *URL:* https://scikit-learn.org/1.5/modules/generated/sklearn.preprocessing.OneHotEncoder.html

- Scikit-learn Developers. "RobustScaler: API Documentation (Stable)." Accessed [October, 2024]. *URL:* https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html