MSc. Data Science and Advanced Analytics 2024/2025

TO GRANT OR NOT TO GRANT: DECIDING ON COMPENSATION BENEFITS

Course: Machine Learning

Professors : Roberto Henriques | Ricardo Santos

GROUP 52:

Alexandre Gonçalves (20240738@novaims.unl.pt)

Diogo Rodrigues (20240512@novaims.unl.pt)

Lena Hermann (20241446@novaims.unl.pt)

Pedro Costa (20222121@novaims.unl.pt)

Santiago Taylor (20240542@novaims.unl.pt)

Github: https://github.com/LenaHermann/ML_Project_Group52

NOVA IMS Information Management School

Index

Abstract

This report presents a multiclass classification model to predict the injury type (Claim Injury Type) for workers' compensation claims submitted to the New York Workers' Compensation Board. Using labeled data from claims assembled between 2020 and 2022, the study analyzes 32 features and creates 22 aditional features. Key preprocessing steps included outlier detection, normalization, encoding, and feature selection. Multiple machine learning models, including Neural Networks, Random Forest, and Gradient Boosting, were evaluated, with the XGBoost model emerging as the most accurate and generalized solution. This model aims to automate claim classification, improving efficiency and accuracy in the WCB's decision-making process.

Keywords: Project, Machine Learning, Predictive Modelling, Neural networks, Decision Trees, Logistic Regression, Instance-Based Learning, Support Vector Machine, Ensemble, Random Forest, Feature Engineering

1. INTRODUCTION

In the year 2022 workplace injury cost employers $ 102,982 million, with private compensations accounting for 60.8% of the total compensation amount [1]. Although the number of work related claims have been declining over the last two decades, average cost per claim has been steadily increasing *(Figure 1)*, showing the importance to optimize and predict insurance claims.

This project focuses on assisting the New York Workers' Compensation Board (WCB) in automating the classification of worker's compensation claims, a process that has traditionally been time-consuming. By replacing manual reviews with automated predictions, the proposed solution seeks to increase the efficiency and consistency of the WCB's decision-making process.

Secondly, by applying supervised and unsupervised machine learning techniques the project focuses on building and optimizing the selected models using techniques such as hyperparameter tuning, feature selection, and preprocessing procedures. Additionally, the project also builds an interface for real-time predictions, and investigates whether incorporating additional variables can further improve model performance.

We started this project by conducting a thorough review of articles whose content has proven highly relevant to our study. Similar projects on worker's compensation claims prediction have identified key predictors like age, injury type, and litigation status, using methods such as support vector machines , regularized logistic regression, neural networks, and other methods.

The paper by *Matthews (2016)* employed Support Vector Machines (SVMs) and regularized logistic regression to improve severity prediction, achieving higher accuracy through class balancing. The author states that SVMs were chosen for their flexibility in parameter tuning and ability to optimize profitability, while logistic regression and random forests faced challenges with high-cardinality variables. In summary, SVMs proved to be the most effective for improving predictions and managing intervention costs.

Additionally, *Tjahjono et al. (2024)* compared tree-based and neural network-based models for worker's compensation claims prediction. The paper found out that tree-based methods, particularly XGBoost, achieved the highest accuracy across multiple datasets, particularly in Mean Absolute Error (MAE) metrics, due to their ability to handle missing values and optimize performance. Neural network-based models like Multi-Layer Perceptron (MLP) and TabTransformer showed mixed results, with MLP occasionally outperforming tree-based methods in some datasets, performing well on R^2 and Root Mean Squared Error (RMSE) metrics. The punchline of the study is the potential of self-attention-based embedding processes, exemplified by TabTransformer, to improve neural network performance by enabling feature transformations that go beyond categorical data.

As our dataset contained a highly imbalanced target variable, we reviewed some literature on the topic. *Tanha et al. (2020)* reviewed 14 boosting ensemble algorithms to address multi-class imbalanced datasets, evaluating them on 19 datasets with varying imbalance ratios. CatBoost

consistently outperformed other methods on conventional datasets across metrics such as MAUC, G-mean, and MMCC, followed by SMOTEBoost, which highlighted the effectiveness of oversampling. XGBoost and LogitBoost also showed strong performance, particularly in multi-class setups without class decomposition. However, boosting methods struggled on highly imbalanced large datasets, with LogitBoost performing best overall.

To conclude our research on imbalanced learning techniques, we explored hybrid resampling algorithms, which combine oversampling and undersampling to address class imbalance by reducing majority samples and increasing minority samples. *Rawat et al. (2022)* presented interesting methods, including SA-CGAN, SMOTified-GAN, and the Hybrid bag-boost model with K-Means SMOTE–ENN. SA-CGAN and SMOTified-GAN, both deep learning-based, use Generative Adversarial Networks (GANs) to generate high-quality synthetic data, excelling in complex, high-dimensional datasets. SA-CGAN improves F1-score by reducing noisy samples and overfitting, though it struggles with unexplored local attributes. SMOTified-GAN balances accuracy and time complexity but operates as an offline pre-processing method, limiting its use for real-time tasks. In contrast, the Hybrid bag-boost model integrates K-Means clustering, SMOTE, ENN, and ensemble techniques to manage noisy binary datasets effectively, though it is not ideal for multi-class problems.

2. Data Exploration , Preprocessing and Feature Engineering

2.1 Exploratory data analysis

When tackling a problem, it's essential to start by understanding the objective and goals. Once that's clear, the next step is to examine the variables provided in the dataset. Gaining an initial understanding of the data can help identify any inconsistencies or patterns, which can make future analysis more efficient.

An example concerning this, is feature selection. Looking at the type of data specifically and categorizing it (either categorical or numerical) allowed us to run several feature selection models and better interpret the results depending on the type of data that we had provided to said models, following the type of target variable.

By using the function ".info()" we got an initial overview of the data types, provided in our dataset. It turns out that we had eight numerical features and twenty five non-numerical features, as depicted in *figure 2*.

2.1.1 Target Variable

Additionally, we found out that our target variable – Claim Injury Type – was highly imbalanced. This variable had eight outcomes, namely "Cancelled" , "Non-Comp", "Med Only", "Temporary",

“PPD SCH Loss“ , “PPD NSL“ , “PTD“ and “Death”, accounting for 2.3%, 41.6%, 13%, 28%, 9%, 0.8 %, 0.01% and 0.08% of the total cases respectively. The last three outcomes are particularly concerning given the fact that they are poorly represented in the whole sample.

The joint distribution of “Age at Injury” and the target *(figure 2)* shows that “NON-COMP” is the most common claim type across all age groups, with claims concentrated among individuals aged 20 to 60, reflecting the working-age population. Claim frequency drops significantly for those under 20 and over 60, with rare types like “Death” and “PTD” occurring infrequently. A notable increase in “NON-COMP” claims for individuals under 20 suggests special cases, anomalies, or youth employment-related claims.

Moreover, regarding the spearman correlation with the target feature (*figure 3* – Correlations), we found out that the strongest correlation is between “Average Weekly Wage” and the target (0.855), indicating a strong relationship. Moderate correlations include “Agreement Reached” (0.236) and “Age at Injury” (0.119).

2.1.2 Independent Variables

After an initial exploration of the independent variables, several observations were made (*figure 11*).

Initially, regarding numerical independent variables: “Age at Injury” variable contains evident outliers, though most claims are concentrated within the working-age population. The “Average Weekly Wage” demonstrates significant variability, with extreme outliers and signs of missing or unreported values. Similarly, “Birth Year” includes unrealistic entries, suggesting errors or inconsistencies in the dataset.

The “IME-4 Count” variable exhibits anomalies, with values far beyond typical ranges, indicating the presence of outliers. In contrast, the “Claim Identifier” displays high variability, which is expected given its nature as a unique identifier. The “Number of Dependents” appears to be within a reasonable range. (figure boxplots)

Regarding the categorical features, the “Alternative Dispute Resolution” and “Attorney Representation “ variables are predominantly labeled as 'N', indicating that dispute resolution and attorney involvement are rare occurrences in the dataset. Similarly, the “COVID-19” dummy variable shows that most claims are unrelated to COVID-19.

The “Carrier Name” feature displays high diversity of names, with "STATE INSURANCE FUND" being the most frequent, while private carriers dominate the “Carrier Type” variable. Claims are most common in Suffolk County, and the NYC District handles the highest volume of cases. The “Gender” variable reveals that the majority of claimants are male.

For industry-related features, the "Health Care and Social Assistance" sector is the most represented in the “Industry Code Description”. The “Medical Fee Region” is dominated by Region IV. Regarding injuries, the most frequent “WCIO Cause of Injury” is "Lifting," the most

common “Nature of Injury” is "Strain or Tear," and the Lower Back Area is the most affected body part.

Lastly, the Zip Code feature has high cardinality, with the most frequent zip code being 11236. All records in the “WCB Decision” variable are labeled as "Not Work Related," showing no variability in this feature.

2.2 Data Preprocessing

Starting off by analysing missing values in the dataset, we’ve found that the “OIICS Nature of Injury Description” variable was entirely missing, and thus it was dropped. Similarly, variables like “IME-4 Count” and “First Hearing Date” had over 70% missing data , low variance and almost no correlation with the target, offering little value to the model and thus dropped as well. Adding on that, the “C-3 Date” variable had a high correlation with other features and 67.4% missing data, making it redundant and it was dropped.

Other variables such as “WCB Decision” had only one value and consequently offered no predictive power to the model, being excluded, as a result. Additionally, the “Alternative Dispute Resolution” feature was dominated by the 'N' category (99.5%) and provided no meaningful insights for rare claim types, making it ineffective as a predictor. Lastly, the “Agreement Reached” variable, unavailable in the test set due to its reliance on post-outcome data, was removed to ensure consistency between training and production environments.

In the dataset, several variables included pairs of codes and descriptions. To optimise the dataset, codes were dropped in favor of descriptions, which were more suitable for grouping and analysis.

For “Industry Code”, “WCIO Cause of Injury”, and “WCIO Part of Body”, each code mapped to a unique description, but some descriptions were linked to multiple codes. Similarly, for “WCIO Nature of Injury”, there was a strict one-to-one relationship between codes and descriptions, making the codes redundant. By prioritising descriptions, the dataset was simplified, reducing dimensionality while retaining interpretability.

To address high cardinality in descriptions, features were mapped into broader categories. The "WCIO Cause of Injury Description", "WCIO Nature of Injury Description", and "WCIO Part of Body Description" columns were transformed into "Cause Injury Category", "Nature of Injury Category", and "Body Part Category", respectively. Any unmapped values were labeled as "Other."

In the "Gender" column, rare entries ("U" and "X") were consolidated into "Other" to ensure consistency across datasets.

Regarding the treatment of the missing values described previously, our team decided to firstly implement a simple approach by imputing the numerical features missing values with their median, and the categorial features with their mode. Besides this approach, the KNN imputer was used to handle missing values for numerical features. However, we observed that this

imputation method did not lead to any improvement in the model's performance. In fact, the F1 Macro Average scores worsened across several models (*figure 16*). This decline in performance could be attributed to the KNN imputer introducing noise or inaccuracies in the dataset by potentially overwriting important patterns present in the original complete data.

2.3 Feature Engineering and Outlier Handling

The feature engineering process involved binning some features into groups and also creating new cyclical date variables. The variable “Age at Injury” was binned into six categories, forming an “Age Group” variable to represent age distribution. Wages were classified into five categories (“Wage Group”) to address high dimensionality caused by numerous unique wage values and a high presence of zero values (214 763 observations). Carrier names with fewer than 5000 occurrences were grouped into an “Other” category for the same reasons stated previously with wages.

For date features, several transformations were applied (*Table 1)* lag times were calculated to measure delays between key dates, such as the difference between Accident Date and Assembly Date (“Lag Time”) and “Accident Date” and “C-2 Date” (“Accident to C2 Lag”). Temporal features like year, month, and day were also extracted from the “Accident Date”, “Assembly Date”, and “C-2 Date”. By using sine and cosine transformations, we created cyclical representations of months, ensuring that December and January were captured as consecutive points. This approach allowed the model to capture temporal patterns. Additionally, the earliest “Accident Date” from the “X_train” dataset was identified and stored as min_accident_date, serving as a reference point for calculating time-based features.

When creating new features, it’s crucial to ensure that those features do not introduce additional noise in the model, and that’s why we decided to treat all the outliers after creating the new features. For this reason, we decided to address all outliers after the new features were created. As shown in *Figure 11,* some features such as “Accident_to_C2_lag,” “Accident_Date_Year,” “C-2_Date_Year,” “Accident_Year,” and “C-2_Year” contained a significant number of outliers. To address these, we adopted the Interquartile Range (IQR) method, a simple yet effective approach, where data points falling outside 1.5 times the IQR from the quartiles were excluded.

For the “Age at Injury” variable, we used a slightly different approach. Rows where the age was below 14 or above 90 were removed, as these ages are either not eligible for the labor market or represent extremely rare cases. The outlier handling process resulted in the removal of approximately 5.6% of the data, which is a reasonable amount and does not pose any significant constraints on the reliance of the model.

Lastly, two new features, 'Region' and 'State', were dynamically engineered from the 'Zip Code' column across multiple datasets to extract geographic information. The 'Region' feature was

derived by extracting the first digit of the zip code, while 'State' used the first two digits, allowing for segmentation of the data at regional and state levels. To ensure consistency, a cleaning process was applied where invalid values in the 'Region' column, such as alphabetic characters or symbols, were replaced with the label 'other', and non-numeric or out-of-range values in the 'State' column were either replaced with 'other' or adjusted to valid integers.

3. Multiclass Classification

3.1 Feature selection

After preparing the data to build the model, we must first perform a feature selection of the variables. The reasons that lead us to make a feature selection were that, it allows the algorithm to train quicker, it diminishes the complexity of the model and makes it simpler to analyse the result, diminishing overfitting and improves the model accuracy if the right set of variables are selected [5].

Our feature selection strategy involved multiple methods tailored to numerical and categorical features to ensure a robust and diverse selection of predictors. For numerical features, we began with correlation analysis, where we calculated Spearman correlations between features and removed one feature from pairs with a correlation exceeding 0.7, reducing multicollinearity. Next, we used Recursive Feature Elimination (RFE) with XGBoost, testing subsets of features ranging from 10 to 14 and selecting the optimal subset based on the highest F1 Macro Score on the validation set. In addition, we applied Random Forest Feature Importance, retaining only features with an importance score above the threshold of 1 / number of features, ensuring that only predictors with above-average contributions were kept. Finally, we applied Lasso Regression, selecting features with coefficients greater than 0 to focus on those with meaningful contributions to the model. For categorical features, we utilized the Chi-Squared Test to evaluate dependency with the target variable, keeping only features with p-values below 0.001 as significant predictors. We complemented this with Cramér's V, retaining categorical features with a strength of association greater than 0.1, ensuring a focus on meaningful relationships.

After applying all these methods, we retained a total of 14 (encoded 45) final features for the model, balancing numerical and categorical predictors while maintaining a diverse and effective set of features.

3.2 Results and Discussion

In the scenario without outlier removal and robust scaling (*Figure 15*), models like Decision Tree and Stacking achieved perfect F1 macro average scores on the training set (1.00). However, this performance reflects significant overfitting, as their validation and test scores were substantially lower. On the other hand, simpler models such as Logistic Regression and Gaussian NB displayed consistent, albeit low, performance across all sets. This outcome highlights their simplicity and inability to capture the complex relationships present in the data. Among all models, XGBoost demonstrated the best balance between training and generalization, with validation and test scores of 0.351, indicating its suitability for unseen data.

With outlier removal, KNN imputation, and robust scaling (*Figure 16*), most models exhibited noticeable improvements in their validation and test scores. Models such as KNN, Bagging Classifier, and XGBoost benefited the most from this preprocessing step. However, Logistic Regression and Gaussian NB maintained their low performance, showing limited adaptability to the dataset. Stacking showed slight improvement compared to its performance in Figure 1 but still struggled to generalize as effectively as XGBoost.

When outlier removal, min-max scaling, and median imputation were applied (*figure 17*), the Neural Network displayed consistent validation and test scores (both at 0.32), representing a slight improvement over its performance with robust scaling. Meanwhile, Random Forest and XGBoost emerged as the top-performing models, achieving validation and test scores of 0.34 and 0.345, respectively. Models with simpler assumptions, like Gaussian NB, remained the least effective, with a notably low test score of 0.10.

Finally, in the scenario without outlier removal, min-max scaling, and median imputation (Figure 4), the Decision Tree again displayed overfitting tendencies, achieving a perfect training score but much lower validation and test scores. XGBoost continued to perform reliably, achieving a score of 0.35 for both validation and test sets, demonstrating its robustness to variations in preprocessing. The Stacking model, however, experienced slight drops in test scores, indicating some sensitivity to the absence of preprocessing.

The Decision Tree and Stacking models frequently displayed perfect training scores (1.00), which is a clear indication of overfitting to the majority class in the dataset. This overfitting was evident in their significantly lower validation and test scores, highlighting their inability to generalize effectively to unseen data.

To address these issues, we incorporated strategies to handle class imbalance in Section 3.3, where we implemented the ADASYN (Adaptive Synthetic Sampling) technique. This method significantly enhances the performance of models on minority classes by generating synthetic samples in a more targeted manner than SMOTE. ADASYN ensures that the model does not disproportionately focus on the majority class, improving its ability to predict rare outcomes effectively.

Additionally, ensemble methods, such as the Bagging Classifier, demonstrated promise when paired with robust preprocessing techniques, as shown in *Figure 15*. Their performance could be further enhanced by fine-tuning hyperparameters, potentially leading to better generalization and improved predictions across all classes.

3.3 Class Imbalance Mitigation with ADASYN

To address the class imbalance in the dataset, we applied ADASYN (Adaptive Synthetic Sampling**)** [6] to oversample the minority classes. This approach generates synthetic samples based on the density distribution of minority class instances, thereby improving class representation and enabling models to learn patterns more effectively.

The results presented in *Figure 19* demonstrate that oversampling with ADASYN led to modest improvements in the validation and test F1 macro scores across most models. Notably, models such as XGBoost and the Bagging Classifier showed slightly higher scores compared to the strategy without sampling. This indicates that oversampling helped address the issue of imbalanced classes to some extent, enabling the models to generalize slightly better to unseen data. However, we also observed that some models still exhibited signs of overfitting, particularly KNN and Neural Networks, which achieved exceptionally high training scores but lower validation and test scores. This suggests that these models may be too complex or sensitive to the training data, despite the use of ADASYN and hyperparameter tuning. In contrast, simpler models such as Logistic Regression and Gaussian Naive Bayes performed more consistently across datasets, albeit at lower overall performance levels.

While the approaches without a sampling strategy resulted in more consistent scores across training, validation, and test sets, the overall performance metrics were slightly lower compared to the ADASYN-based approach. This highlights the trade-off between addressing class imbalance and avoiding overfitting.

4. Open Ended Section

Initially, to adress the open ended section we compared the performance of the best-performing model, XGBoost, with a stacking ensemble approach. Stacking combines predictions from multiple base models to produce a more robust and generalized output by leveraging their individual strengths.

The stacking ensemble was built using Naive Bayes, Decision Trees, XGBoost, Logistic Regression, and SVM as base models, with a Multi-Layer Perceptron (MLP) classifier serving as the meta-model.

While the stacking ensemble showed a slight improvement over XGBoost on the test set, the performance gain was marginal. This suggests that stacking can reduce overfitting and capture diverse patterns, but it comes with higher complexity and runtime costs.

Overall, XGBoost remains a strong candidate due to its efficiency and interpretability through feature importance scores. The stacking ensemble offers an alternative for scenarios where minor performance gains justify additional computational overhead.

Additionally to address the second part of the open ended section of the project, we deployed the trained insurance claim prediction model as a live web application using Streamlit, making it accessible to end-users in an intuitive and interactive way. The website can be found live at: https://ml-project-deployment-g52.streamlit.app/

The deployment involved connecting the trained model and preprocessing pipeline into a single file, which was then integrated into a Streamlit-based interface. This interface *(Figure 20)* allowed users to input claim-related details, such as age, average weekly wage, and number of dependents, and receive real-time predictions of the claim outcome. The app displayed user inputs alongside predictions for transparency and ease of use *(Figure 21).* By hosting the application online, we ensured it was publicly accessible without requiring any local setup. Moreover, this deployment not only demonstrated the model's practicality but also highlighted its potential to support decision-making in real-world insurance scenarios, applying machine learning development to end-user applications.

5. Conclusion

After performing the analysis regarding the predictive modelling for the classification claim injury type on our claim's dataset, there could be some improvement points for future analysis. More specifically, concerning the original dataset, it is an imbalanced one, which can be problematic for classification. Concretely, "standard classifiers tend to be overwhelmed by the large classes and ignore the small ones". *[7]*

Actually, there are many re-sampling techniques available for imbalanced datasets, and some of them were tested in this project, including Adaptive Synthetic Sampling (ADASYN). While ADASYN showed better performance with models like XGBoost and the Bagging Classifier, it also introduced some problems, particularly overfitting on the training data. With ADASYN, XGBoost improved its test F1 Macro Average score to 0.37, compared to 0.351 achieved with earlier preprocessing strategies without sampling. Similarly, the Bagging Classifier achieved a test score of 0.35, slightly outperforming its previous result of 0.346 without sampling. This is a good point for improvement in the future since it would be beneficial to have a model that predicts as well for balanced datasets as well as good for imbalanced ones, indeed it would be the ideal case.

Therefore, a deep exploration in the matter of re-sampling techniques for imbalanced datasets as well as their hyper-parameters and parameters would be auspicious for greater performance of the model, both in the training and validation sets.

Besides, the F1-Score in the Kaggle platform was not as good as in the Jupyter Notebook probably because our model was trained based on the unbalanced dataset that predicts better the majority class. Also, the model that we selected as the best one was not the highest score in Kaggle; however, we considered that the elected one was better in terms of performance since Kaggle only shows the result for 30% of the data and a higher F1-Score in Kaggle could be prone to overfitting.

6- References

[1] https://www.nasi.org/research/workers-compensation/workers-compensation-benefits-costs-and-coverage-2022-data/

[2] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10675747

[3] https://jewlscholar.mtsu.edu/server/api/core/bitstreams/37e991bf-e902-4f55-8fb9-dd2998b98830/content

[4]https://www.researchgate.net/publication/220520061_Editorial_Special_Issue_on_Learning_from_Imbalanced_Data_Sets

[5] https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning

[6]https://imbalanced learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html

[7] https://www.researchgate.net/publication/220520061_Editorial_Special_Issue_on_Learning_from_Imbalanced_Data_Sets

7 - Appendix

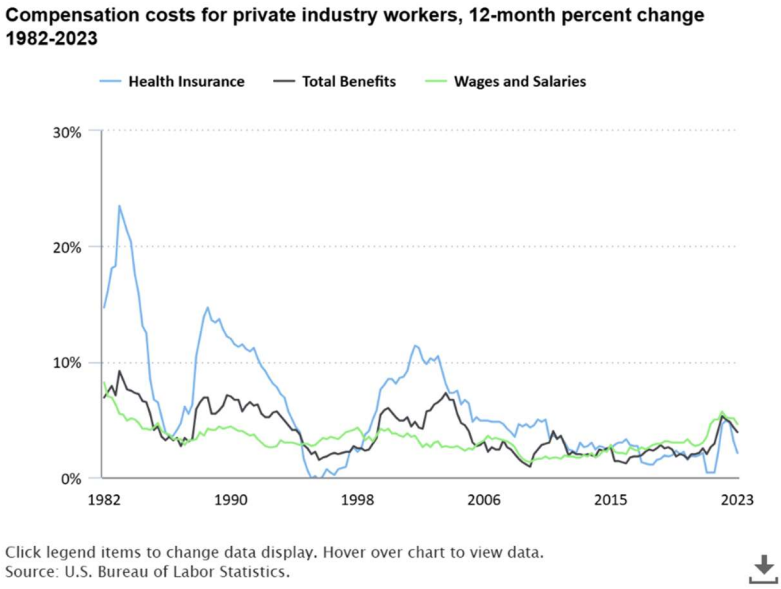

Figure 1 – Compensation costs for private industry workers

#	Column	Non-Null Count	Dtype
0	Accident Date	570337 non-null	object
1	Age at Injury	574026 non-null	float64
2	Alternative Dispute Resolution	574026 non-null	object
3	Assembly Date	593471 non-null	object
4	Attorney/Representative	574026 non-null	object
5	Average Weekly Wage	545375 non-null	float64
6	Birth Year	544948 non-null	float64
7	C-2 Date	559466 non-null	object
8	C-3 Date	187245 non-null	object
9	Carrier Name	574026 non-null	object
10	Carrier Type	574026 non-null	object
11	Claim Identifier	593471 non-null	int64
12	Claim Injury Type	574026 non-null	object
13	County of Injury	574026 non-null	object
14	COVID-19 Indicator	574026 non-null	object
15	District Name	574026 non-null	object
16	First Hearing Date	150798 non-null	object
17	Gender	574026 non-null	object
18	IME-4 Count	132803 non-null	float64
19	Industry Code	564068 non-null	float64
20	Industry Code Description	564068 non-null	object
21	Medical Fee Region	574026 non-null	object
22	OIICS Nature of Injury Description	0 non-null	float64
23	WCIO Cause of Injury Code	558386 non-null	float64
24	WCIO Cause of Injury Description	558386 non-null	object
25	WCIO Nature of Injury Code	558369 non-null	float64
26	WCIO Nature of Injury Description	558369 non-null	object
27	WCIO Part Of Body Code	556944 non-null	float64
28	WCIO Part Of Body Description	556944 non-null	object
29	Zip Code	545389 non-null	object
30	Agreement Reached	574026 non-null	float64
31	WCB Decision	574026 non-null	object
32	Number of Dependents	574026 non-null	float64

Figure 2 – Datatypes by feature

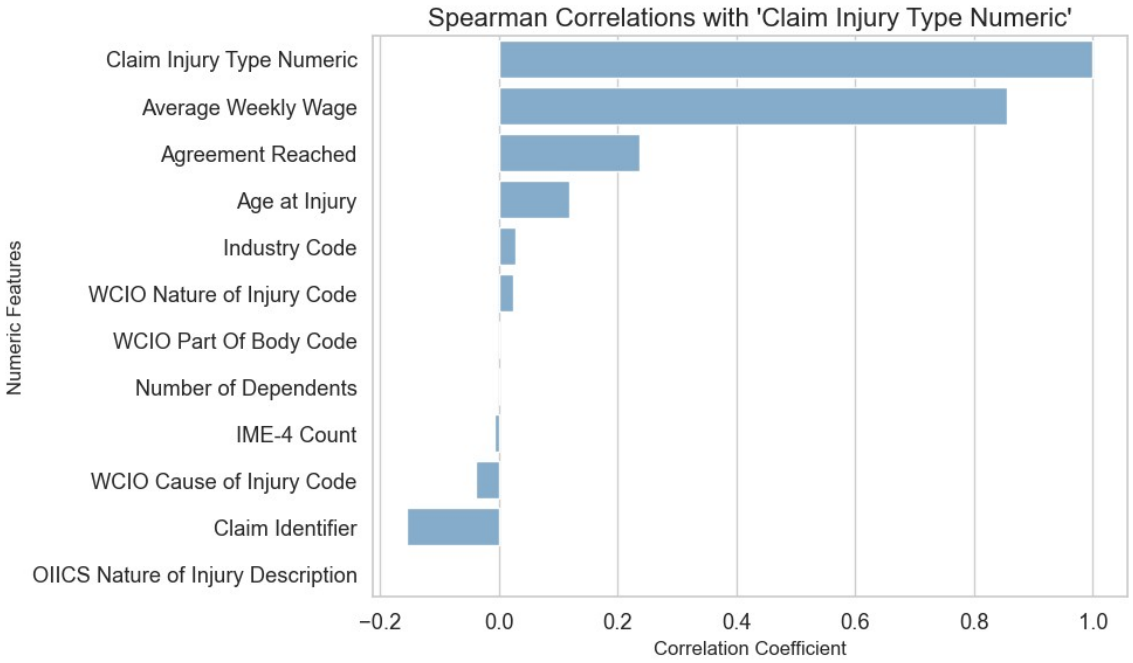

Figure 3 - Spearman Correlations with 'Claim Injury Type Numeric'

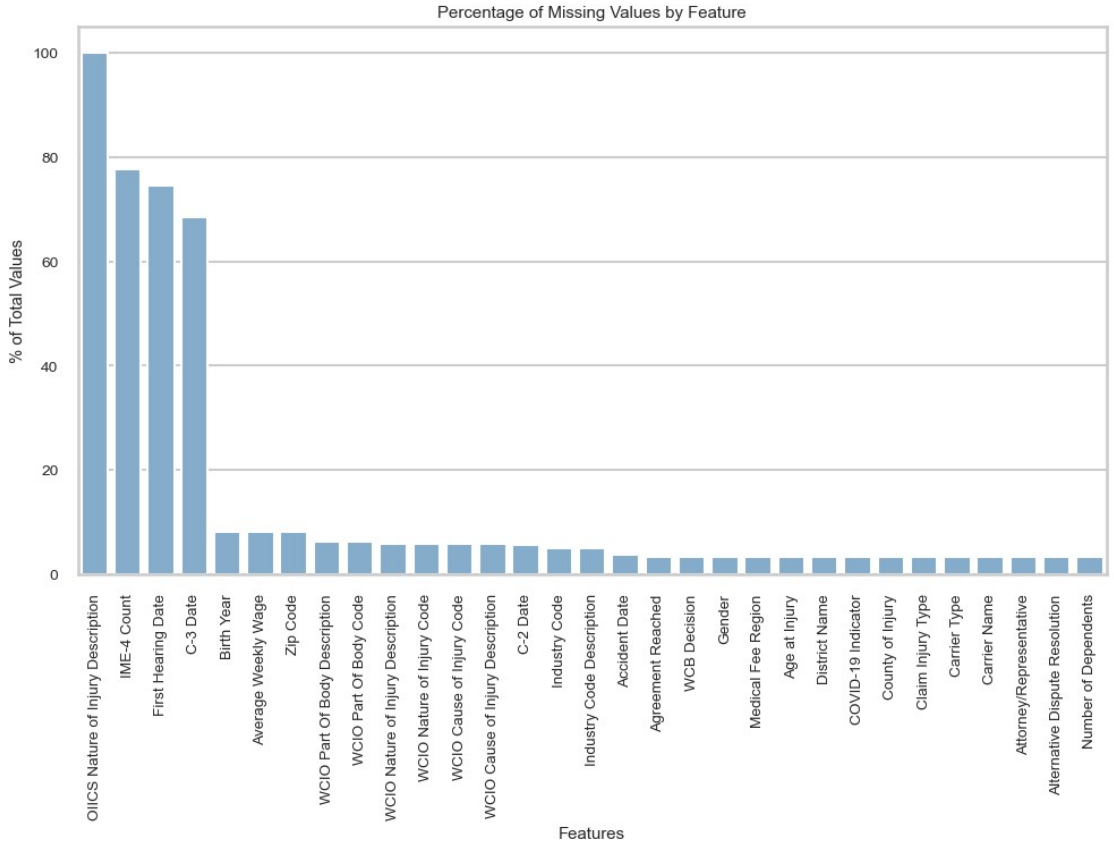

Figure 4 - Percentage of Missing Values by Feature

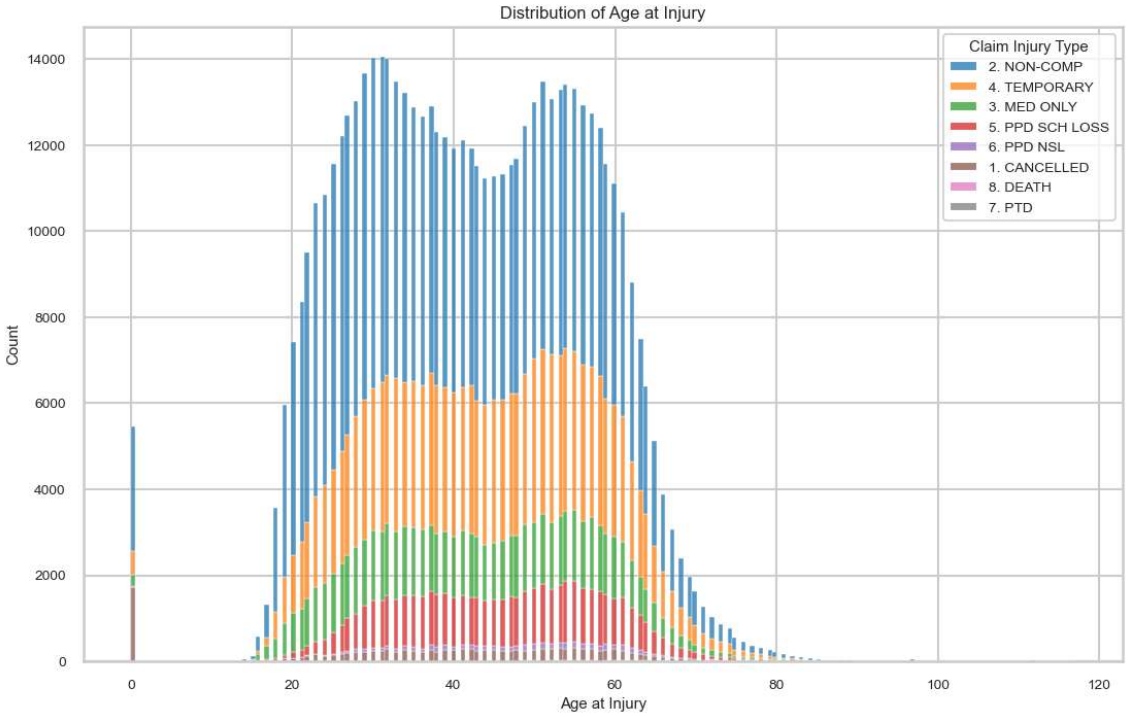

Figure 5 - Distribution of Age at Injury'

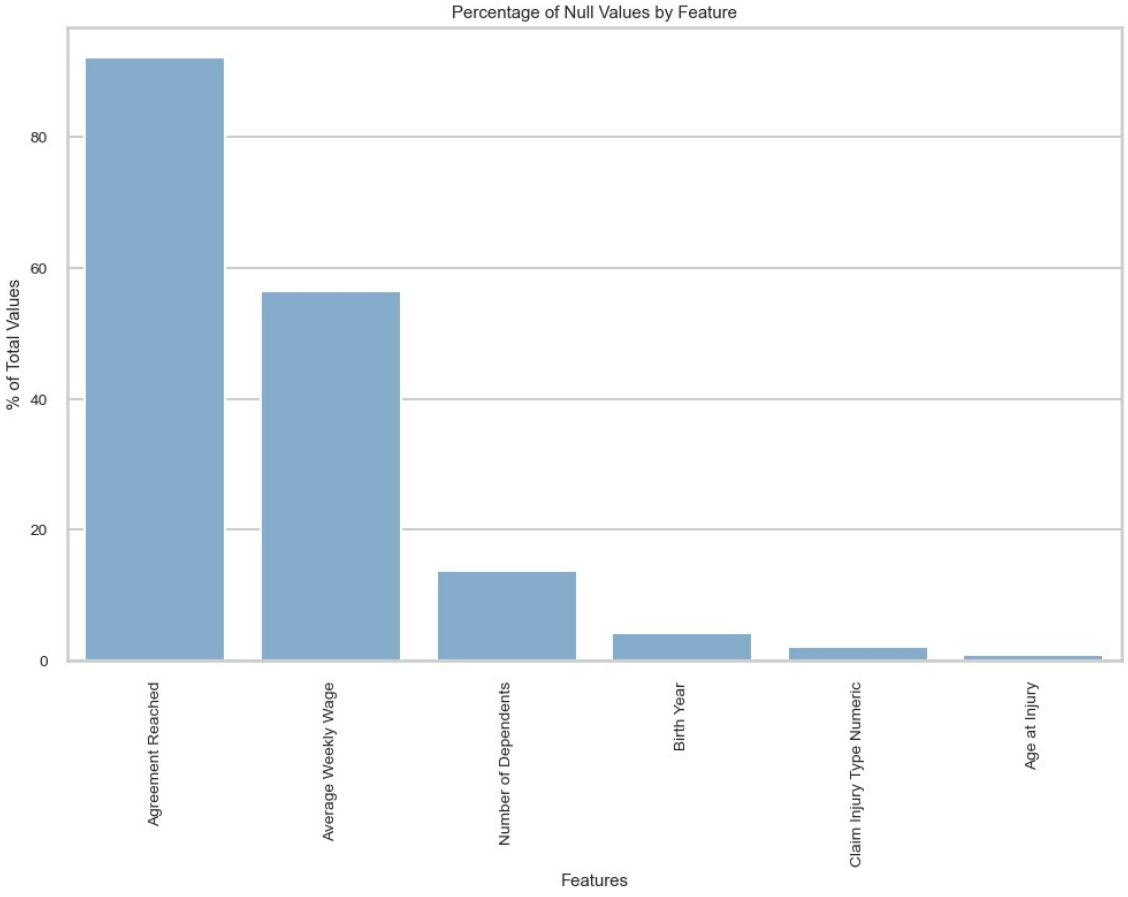

Figure 6 - Percentage of Null Values by Feature'

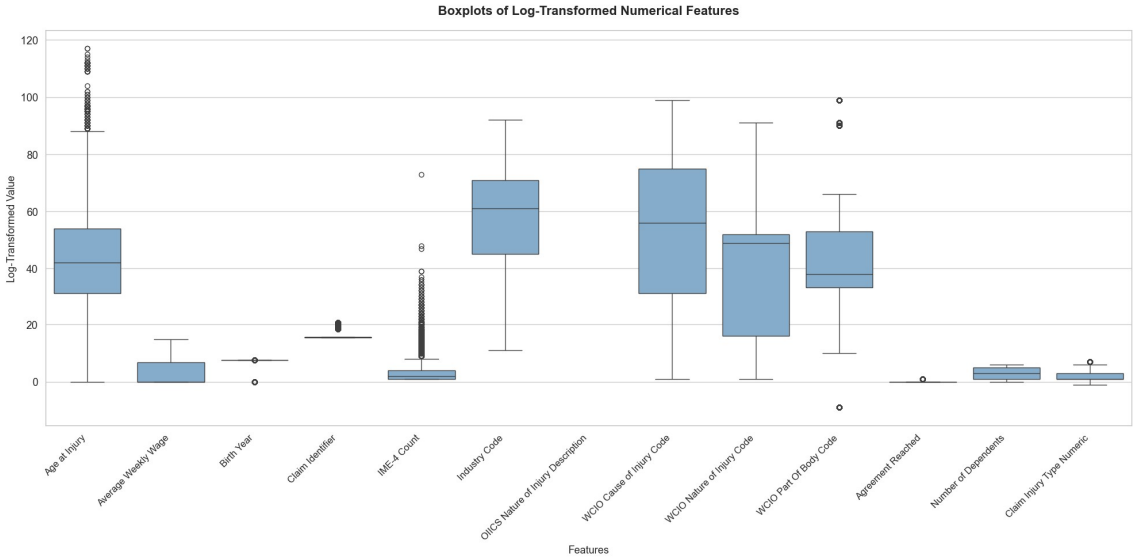

Figure 7 - Boxplots of Log-Transformed Numerical Features

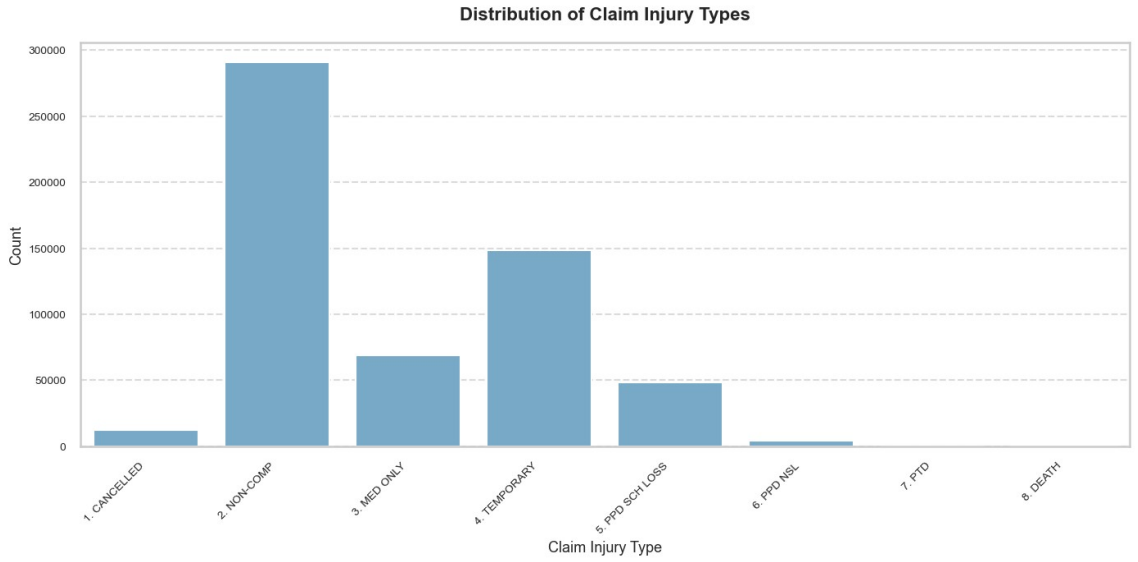

Figure 8 - Distribution of Claim Injury Types

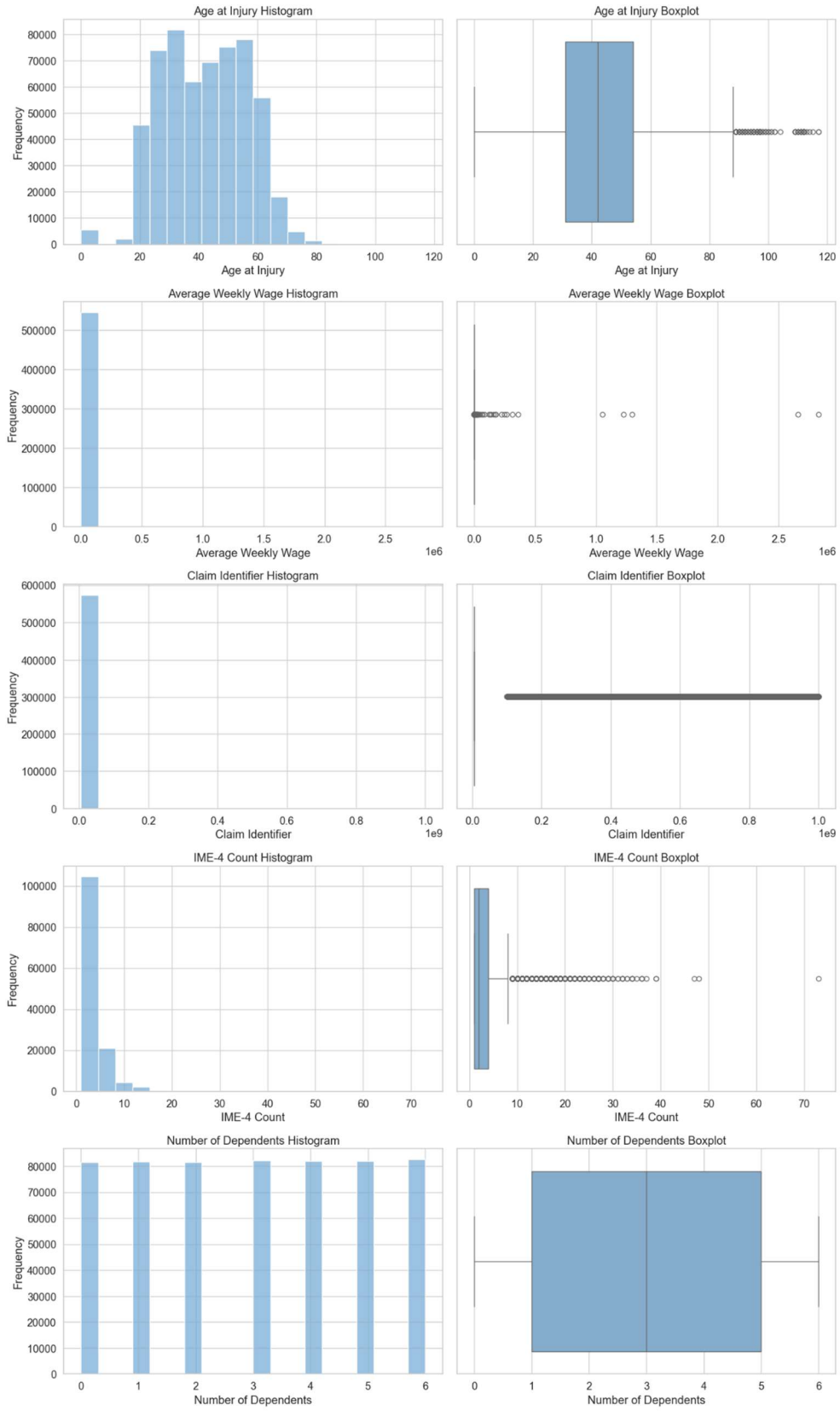

Figure 9 - Analysis of "Age at Injury, Average Weekly Wage, Claim Identifier, IME-4 Count and Number of Dependents"

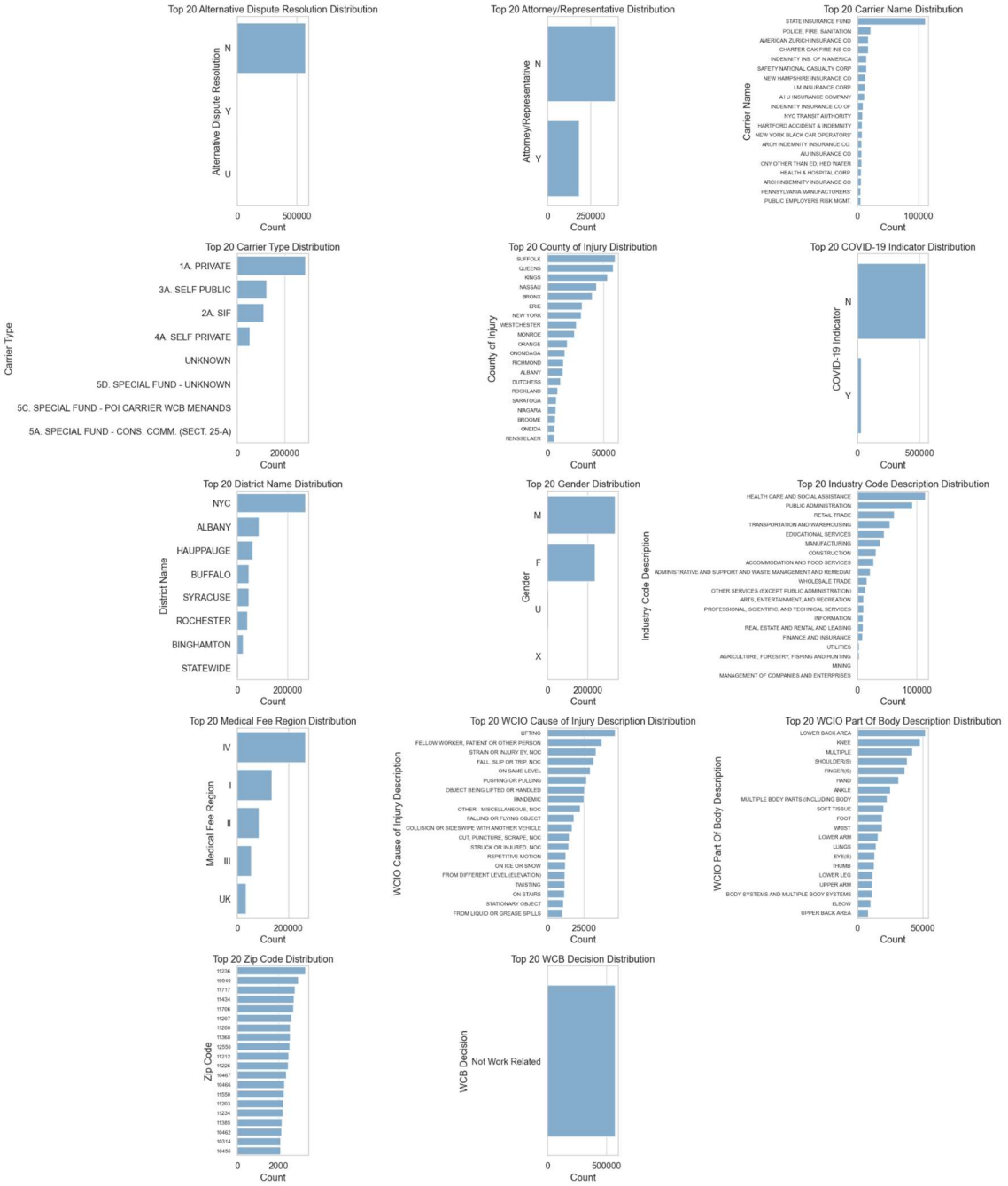

Figure 10 - Visual Data Analysis for the Categorical Features

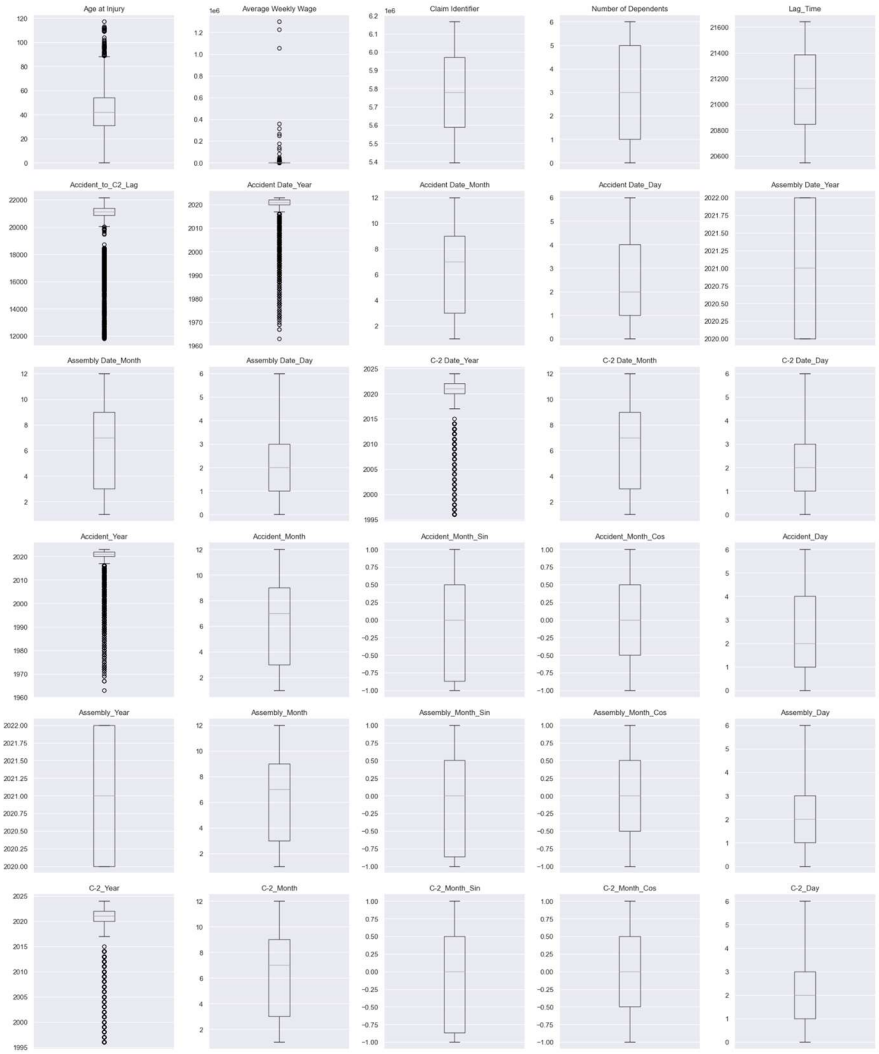

Figure 11 – Boxplots of the numerical features

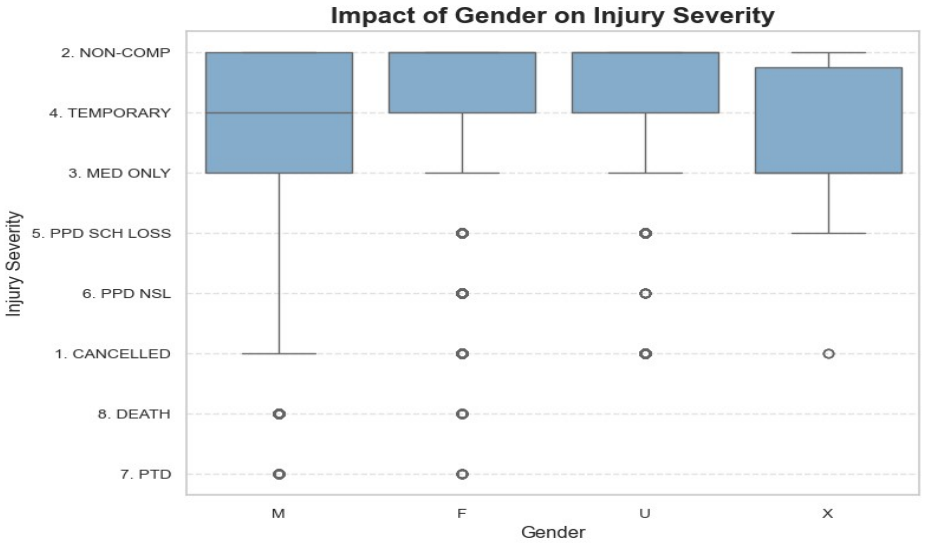

Figure 12 – Boxplots of claim's by gender

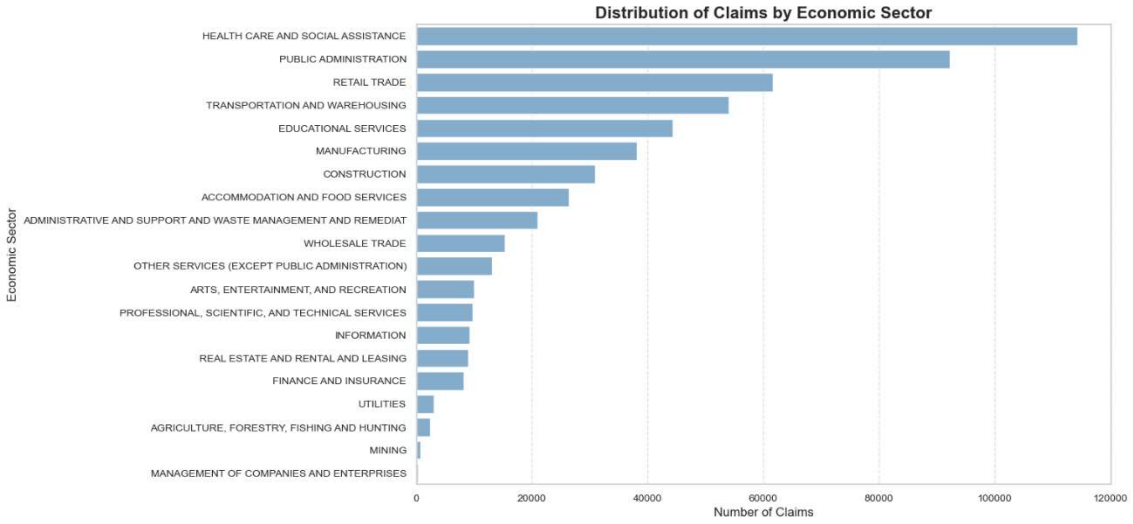

Figure 13 – Distribution of claims by economic sector

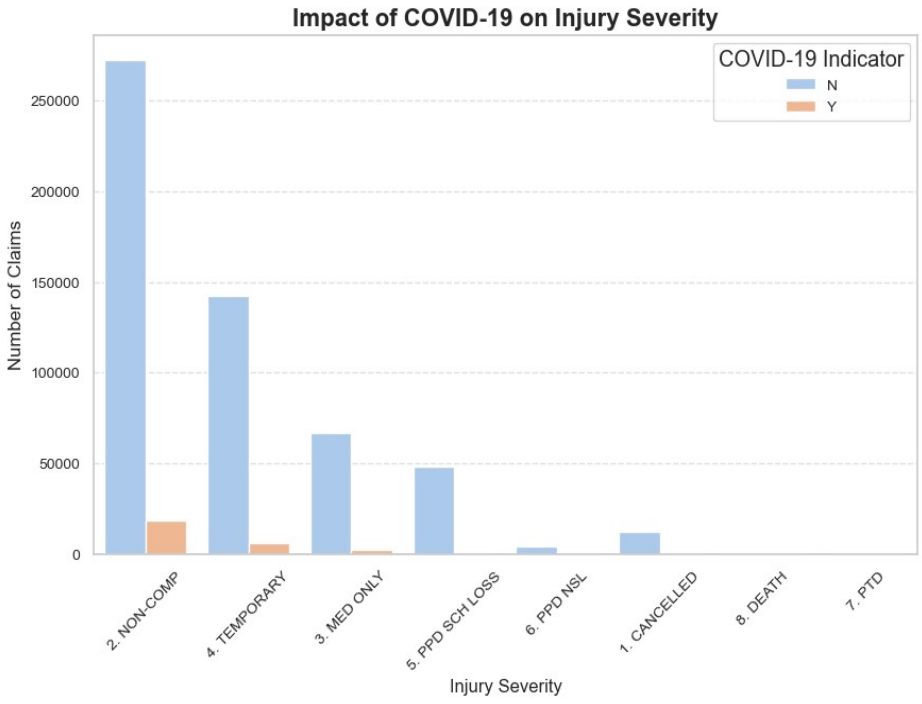

Figure 14 – Distribution injuries in relation to COVID-19 indicator

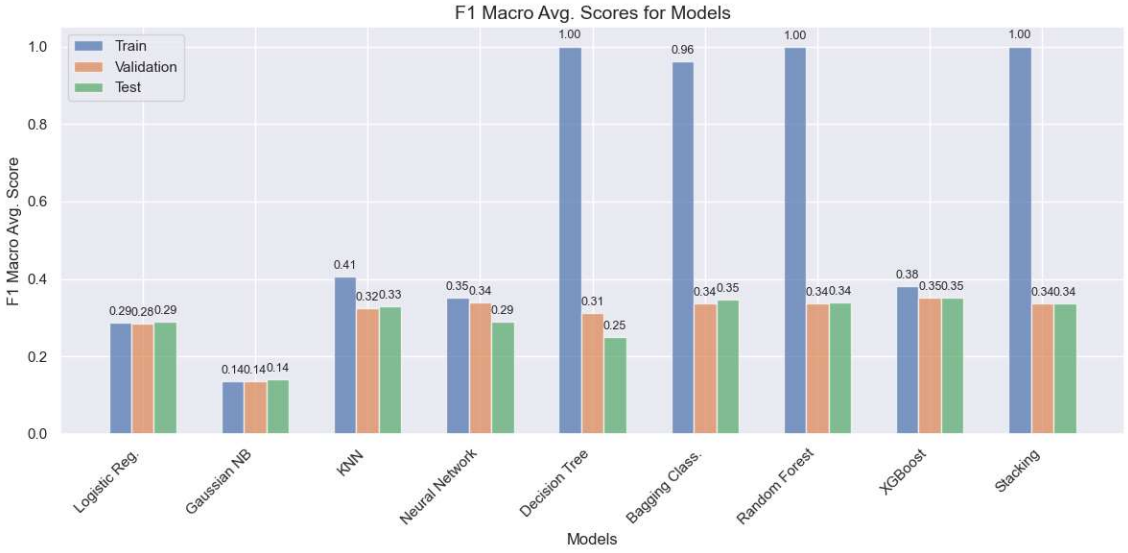

Figure 15 – Models without outlier removal and robust scaler

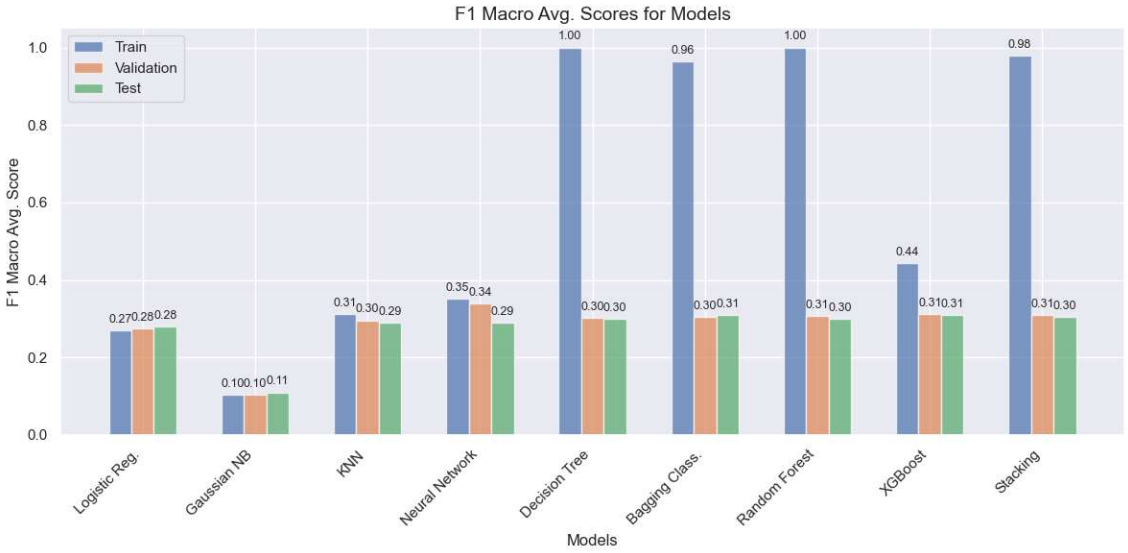

Figure 16 – Models with outlier removal , imputation of missing values using KNN and robust scaler

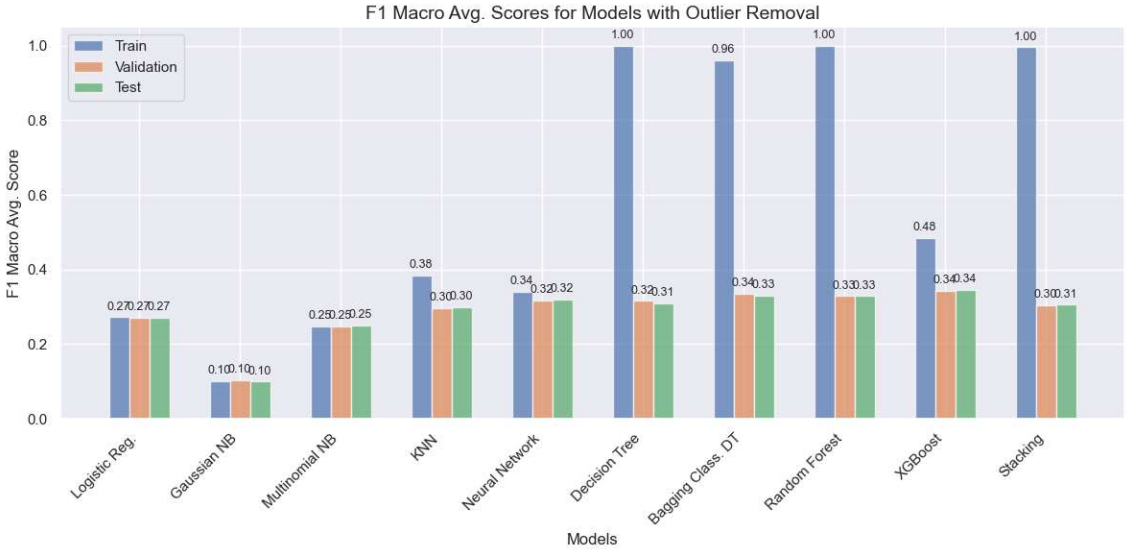

Figure 17 – Models with outlier removal and minmax scaler and median imputation of NaN values

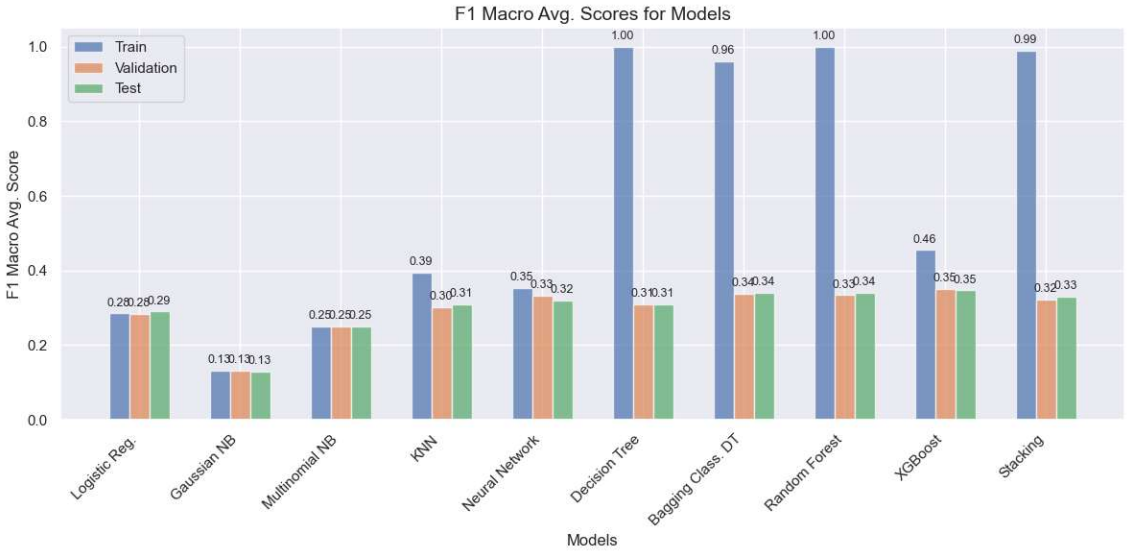

Figure 18 - Models without outlier removal, minmax scaler and median imputation of NaN values

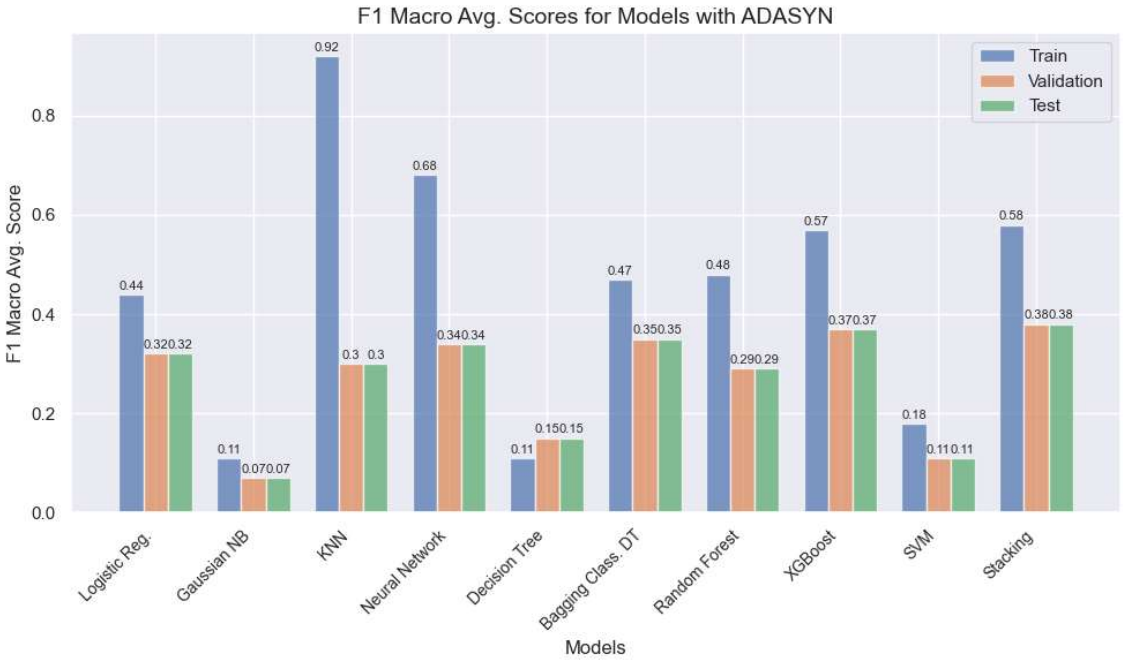

Figure 19 – Models with outlier removal, standard scaler and sampling using ADASYN

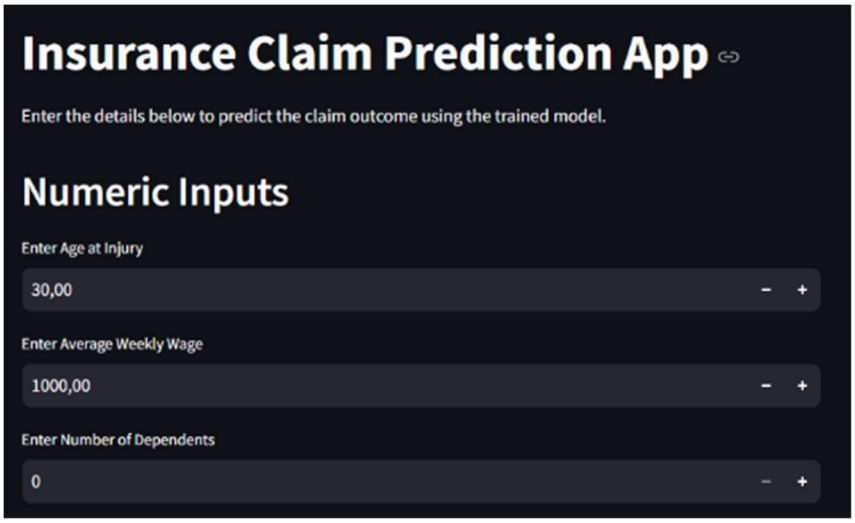

Figure 20 – Web Application user interface

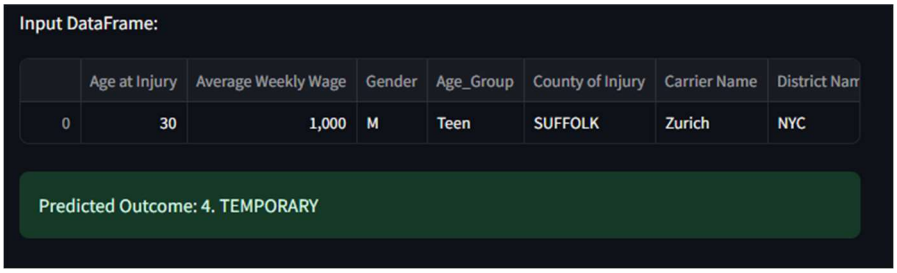

Figure 21 - Feature that show in real time the input value introduced by the user

Table 1 - Feature Engineering

Variable	Formula	Explanation
Lag_Time	(df['Assembly Date'] - min_accident_date).dt.days	Days between the accident date and the assembly date.
Accident_to_C2_Lag	(df['C-2 Date'] - min_accident_date).dt.days	Days between the accident date and the C-2 date.
Accident_Year	df['Accident Date'].dt.year	Year extracted from the accident date.
Accident_Month	df['Accident Date'].dt.month	Month extracted from the accident date.
Accident_Month_Sin	np.sin(2 * np.pi * df['Accident_Month'] / 12)	Sine transformation of the month to capture its cyclic nature.
Accident_Month_Cos	np.cos(2 * np.pi * df['Accident_Month'] / 12)	Cosine transformation of the month to complement the sine feature.
Accident_Day	df['Accident Date'].dt.dayofweek	Day of the week extracted from the accident date (0 = Monday).
Assembly_Year	df['Assembly Date'].dt.year	Year extracted from the assembly date.
Assembly_Month	df['Assembly Date'].dt.month	Month extracted from the assembly date.
Assembly_Month_Sin	np.sin(2 * np.pi * df['Assembly_Month'] / 12)	Sine transformation of the month to capture its cyclic nature.
Assembly_Month_Cos	np.cos(2 * np.pi * df['Assembly_Month'] / 12)	Cosine transformation of the month to complement the sine feature.
Assembly_Day	df['Assembly Date'].dt.dayofweek	Day of the week extracted from the assembly date.
C-2_Year	df['C-2 Date'].dt.year	Year extracted from the C-2 date.

C-2_Month	df['C-2 Date'].dt.month	Month extracted from the C-2 date.
C-2_Month_Sin	np.sin(2 * np.pi * df['C-2_Month'] / 12)	Sine transformation of the month to capture its cyclic nature.
C-2_Month_Cos	np.cos(2 * np.pi * df['C-2_Month'] / 12)	Cosine transformation of the month to complement the sine feature.
C-2_Day	df['C-2 Date'].dt.dayofweek	Day of the week extracted from the C-2 date.
Age_Group	pd.cut(df['Age at Injury'], bins=age_bins, labels=age_labels)	Binned age groups for the individual (e.g., 20-29, 30-39, etc.).
Wage_Group	pd.cut(df['Average Weekly Wage'], bins=wage_bins, labels=wage_labels)	Binned wage groups (e.g., Zero, Below Average, High).
Carrier Name Grouped	df['Carrier Name'].apply(...)	Groups carrier names with low counts into an "Other" category.
Region	df['Zip Code'].astype(str).str[0]	Extracts the first digit of the zip code to classify geographic areas into broad regions.
State	df['Zip Code'].astype(str).str[:2].apply(lambda x: 'other' if not x.isdigit() or int(x) > 99 else int(x))	Extracts the first two digits of the zip code to identify the state. Invalid values are replaced with "other".