TO GRANT OR NOT TO GRANT: DECIDING ON COMPENSATION BENEFITS

Machine Learning Group Project **(Fall Semester – 2024/2025)**

Nova Information Management School

Master's in Data Science and Advanced Analytics

GROUP 29

Inês Honrado, 20240559

Jude Gbenimako, 20240700

Ruben Marques, 20240352

Susana Reis, 20240567

Tomás Carvalho, 20240938

Contents

Group Member Contributions 2
Abstract 2
Introduction 3
Data Exploration and Preprocessing 3
Multiclass Classification 6
Feature Selection 6
Model Comparison and Selection 7
Model Improvement 9
Extra Preprocessing 10
Open-Ended Section 10
Conclusion 11
Annexes 12
References 21

Group Member Contributions

Our approach to this group project was very collaborative. We divided tasks by all team members and monitored progress regularly with scheduled meetings. Jude started by researching similar works to provide inspiration and context for our project. At the same time, Inês was responsible for the preliminary data analysis and preprocessing, although the decisions on how to handle missing data and outliers were made collectively during team discussions. Susana then focused on the feature selection process, with the goal of finding the most relevant variables for the model.

The model development phase was a collaborative work between Jude, Tomas, and Ruben. They evaluated and tested different algorithms to select the most effective final one. Inês then worked to improve the selected model and develop the auxiliary 'Agreement Reached' model. Finally, Susana and Inês worked together to develop the interface for the open-ended section of the project.

Regarding report writing, we also divided the sections based on each member's area of contribution, so the paper reflected our individual efforts, while still maintaining a cohesive narrative.

Abstract

This project aimed to develop an efficient and user-friendly platform for the Workers' Compensation Board (WCB) staff to predict the Claim Injury Type for workers. The main goal was to facilitate decision-making by automating the classification of injury types based on historical claims data. To achieve this, we implemented a machine learning solution using a variety of preprocessing techniques, including handling missing data, outliers, and inconsistent values, followed by feature selection to reduce the number of variables without compromising the model's quality. We explored multiple machine learning algorithms, from traditional models like DecisionTreeClassifier to more complex ones such as neural networks and ensemble methods. After testing these models, the StackingClassifier proved to be the most effective model, and it was then improved, achieving a 0.7255 F1 Weighted score.

To make the model accessible to non-technical users, we developed a platform using Streamlit. This interface allows WCB staff to input features related to a worker's injury and receive immediate predictions along with a confidence score, as well as feature importance information, all derived from the trained model. Despite the promising performance of the final model, some challenges were faced, including the imbalanced target variable and the large dataset size. Future work may, therefore, be done with advanced techniques for balancing the target variable and further refining the model to improve its performance in minority classed. Overall, this project successfully meets its objective of providing WCB staff with an effective, easy-to-use tool to assist in the claims process, and demonstrates the potential of machine learning in improving operational efficiency.

Everything done in the context of this project is public published on GitHub: https://github.com/inesph/Machine-Learning-Group-Project---Group-29.

Introduction

Workers' compensation for workplace injuries in New York is administrated and regulated through the Workers' Compensation Board (WCB). Having manually processed over 5 million claims since the year 2000, the process has become increasingly resource-intensive and very time-consuming due to the complexity and volume of these cases. To solve this problem, **WCB seeks a process that will automate the decision-making for new claims**. Therefore, the **purpose** of this project is to apply Machine Learning (ML) models, taking claims from 2020 to 2022, to conduct automated claim evaluations aimed at speeding up decision-making for new claims and enhancing WCB operational efficiency in handling compensation processes.

Machine learning has been used as a powerful tool in automating decision-making processes across various disciplines, including occupational injury severity prediction and classification. There are **some examples** that showed a transformation and promising approach, such as what was presented by ***Sarkar et al., 2018***. They were able to show the effectiveness of ML algorithms by using optimized Support Vector Machines and Artificial Neural Networks in predicting workplace incidents. They achieved a prediction accuracy of 95%, along with interpretable rule-based insights, and this approach was successfully implemented in industries like steel manufacturing. ***Kakhki et al. (2019)*** applied Boosted Trees and Naïve Bayes models to classify injury severity in agribusiness industries, achieving predictive accuracies as high as 98%. The study identified injury type and worker demographics as some of the most important factors that could enhance safety interventions. In addition, the models optimized by ***Khairuddin et al. (2022)***, such as Gradient Boosting and Random Forest, have identified critical predictive features, including injury type and event conditions. These methods have consistently exceeded 90% accuracy, which can already demonstrate their utility in workplace safety initiatives.

These past works inspired us and motivated us to accomplish the goal of creating a good and solid solution for WCB workers. This report will guide you, as a reader, to understand the **steps taken** until that solution was found. These steps include an initial analysis and preprocessing of our datasets, a cohesive feature selection analysis, model testing and comparison, and the final model improvement.

While previous works used accuracy to measure their performance, we will be applying a **F1 Weighted score metric**, for reasons further explained, and we look forward to achieve at least a **0.65 score**. Additionally, we have the ambition of **making our solution available to everyone**, including those with poor technical knowledge. To address this point, we will show you how we implemented an easy-to-operate and user-friendly platform to facilitate the work of WCB staff.

Data Exploration and Preprocessing

The **preliminary analysis** of the training dataset - *df_train* - led us to find 593,471 observations with 33 features, both numerical and categorical *(Figure 1)*. Most of the variables were substantially populated, but there were **high levels of missing data** for fields such as *C-3 Date*, *IME-4 Count*, and *First Hearing Date*. In fact, the column *OIICS Nature of Injury Description* was completely null. We were also able to recognize *Claim Identifier* as the perfect column to use as **index**, so we right away set it as so for both *df_train* and *df_test*.

With further exploration, we identified **18,350 duplicate rows** in the train dataset that needed cleaning. In addition, **statistical checks** revealed several potential issues: impossible values in *Age at Injury* (from 0 to 117); too many zeros in *Average Weekly Wage*; and a highly skewed mean *Birth Year* of approximately 1886, which was likely influenced by invalid entries. We also noticed that most values in *Agreement Reached* were zeros, indicating that the majority of claims did not reach an agreement, while the *WCB Decision* column contained only a single value, which seriously limited its predictive utility. A **Spearman correlation analysis** revealed a strong negative correlation (-) between *Age at Injury* and *Birth Year*, reflecting their intrinsic (and predictable) dependency. To avoid multicollinearity, one of them will be removed later on. We can suspect that, because of the relevance to worker recovery and risk factors, *Age at Injury* will be more informative, but final decisions will be made on feature selection.

Before diving in outliers and missing values, the pre-processing first involved converting all date-related fields into datetime format for effective manipulation and feature extraction from these variables. After that, null values in the target variable - *Claim Injury Type* - were removed from df_train since they would not contribute to anything in the training of our model. Following this, the dataset was **split into training and validation** sets using the **holdout method** *(Kumar, 2023)*, which facilitates model evaluation on unseen data. The split allocated 80% of the data for training and 20% for validation (test_size=0.2). Besides that, the following parameters have been set to ensure good data handling: random_state=42 for reproducibility, stratify=y to make sure the target variable is well segmented in both sets, and shuffle=True to avoid any bias by first shuffling the data before splitting.

In order to handle the **outliers** in numeric variables, we started by plotting boxplots and trying the **IQR method**. However, this approach proved unsuitable for certain variables, such as *Average Weekly Wage*, where the calculated bounds (e.g., Lower Bound: -1265.01, Upper Bound: 2108.35) seemed unrealistic. Instead, we adopted a **percentile-based approach**, using the 3rd and 97th percentiles for variables *Average Weekly Wage* (leading to more reasonable bounds: Lower Bound: 0.0, and Upper Bound: 2288.68) and *IME-4 Count*. For datetime variables, including *Accident Date*, *Assembly Date*, *C-2 Date*, *C-3 Date*, and *First Hearing Date*, we used the 1st and 99th percentiles to set thresholds for outliers.

In the case of *Age at Injury* and *Birth Year*, **domain knowledge** helped us decide on appropriate bounds. In the workplace context, we recognized ages less than 16 and greater than 85 years as invalid or outliers.

All of those outliers, all detected with the *X_train* based thresholds to **avoid data leakage** into the validation and test set, were **transformed into null values** to be treated along with the missing data. The treatment of outliers becomes evident from the boxplots comparing distributions before and after outlier treatment for the variables *Age at Injury*, *Birth Year*, *Average Weekly Wage*, and *IME-4 Count (Figures 2 and 3)*.

Many of the variables contained both a "code" and a corresponding "description" that represented the same information. Before addressing missing values, we simplified the dataset by **removing redundant "description" variables**. We did so by storing the descriptions in dictionaries that map each code to its corresponding description. This way, we would not lose the descriptive context but could safely drop the "description" columns to keep the dataset more efficient for analysis.

To handle **missing values**, different imputation strategies were applied *(Gelman & Hill, 2006)* for each variable:

- ***Accident Date***: Given the skewed distribution, missing values were replaced by the **median** calculated from *X_train*, so that no bias is created.
- ***Age at Injury***: The median was chosen for imputation.
- ***Alternative Dispute Resolution***: Even though there were no missing values, there were values labeled as 'U' (Unknown). Those were replaced with the **mode** determined from *X_train*.
- ***Assembly Date***: Since there were no missing values in *df_test*, rows with missing values in *X_train* and *X_val* were **dropped**.
- ***C-2 Date***: Missing values were imputed by using a **backfill method** *(Handling Missing Data in Data Pre-Processing, n.d.)* on the data sorted by *Accident Date* so that the temporal relationship between the accident and filing of the C-2 form was maintained.
- ***C-3 Date***: Missing values in *C-3 Date* were treated as cases where the form was not received. A **binary variable**, ***C-3 Received***, was created to indicate the presence (1) or absence (0) of the form.
- ***First Hearing Date***: Missing values in *First Hearing Date* were treated as cases where no hearing had occurred. A **binary variable**, ***First Hearing Happened***, was created to represent whether a hearing date existed (1) or not (0).
- ***Industry Code***: Missing values were imputed using the most frequent *Industry Code* associated with each *Carrier Type* in *X_train*.
- ***Average Weekly Wage***: Missing values were filled based on hierarchical means calculated from *X_train*: a combined mean of *Industry Code* and *Carrier Type* when both were available, followed by individual means for either variable if only one was present, and the overall mean as a fallback.
- ***Birth Year***: Missing values were **calculated** using *Age at Injury* and *Accident Date*.
- ***IME-4 Count***: Based on the observed distribution in *X_train*, we assumed missing values to represent claims without Independent Medical Examination reports and were **imputed as 0**.
- ***Medical Fee Region***: Records labeled as 'UK' were filled with the most frequent *Medical Fee Region* corresponding to the *County of Injury*, determined from *X_train*.
- ***OIICS Nature of Injury Description***: This column was entirely null and was **removed** due to its lack of utility.
- ***WCIO Cause of Injury Code***, ***WCIO Nature of Injury Code***, ***WCIO Part of Body Code***: Missing values in these fields were imputed using some **group-based strategies** derived from *X_train*, considering related variables such as *Carrier Type* and *Injury Code*.
- ***Zip Code***: We noticed that **inconsistent formatting** (floats vs. integers) led to multiple categories for the same zip code. To fix that, we created a function to standardize zip codes, and missing values were filled using **group-based imputation** based on *County of Injury* and *Medical Fee Region* from *X_train*.
- ***WCB Decision***: As this column contained only a single value ("Not Work Related") across all rows in *X_train*, it was **removed** due to its lack of predictive value.

Again, it is important to note that, by strictly deriving imputation values from *X_train*, we made sure that no information from *X_val* or *df_test* runned into the decision of preprocessing, to completely **avoid data leakage** and preserve the validity of validation and test sets for model evaluation.

Next, we engineered more features with the goal of increasing the predictive power of our future model *(Liu, 2018b)*. We extracted the weekday from the Accident Date to create a new feature called ***Injury Day of Week*** *(figure 4)*. We also calculated ***Time to Assembly*** *(figure 5)*, defined as the difference in days between the Accident Date and the Assembly Date, and ***Time to C-2*** *(figure 6)*, defined as the difference in days between the Accident Date and the C-2 Date. These two features were replaced with the mean derived from the training set in case of any negative values. A new feature ***Wage/Dependents Ratio*** *(figure 7)* was created by dividing the Average Weekly Wage by the Number of Dependents. To capture additional insights, we created a binary flag, ***Multiple Form Submissions*** *(figure 8)*, which indicates whether the IME-4 Count is above 1 and ***Above County Wage Average*** *(figure 9)*, which identifies whether a row's wage is above the average for its *County of Injury*.

Before encoding and scaling, we transformed *Alternative Dispute Resolution*, *Attorney/Representative* and *COVID-19 Indicator* into **binary variables**, and we **fixed data types** across the datasets using a custom function to cast variables into their appropriate types. Also, since some machine learning models do not handle datetime variables natively, we converted these columns into a numerical variable transforming them into an **integer of the year-month-day (YMD) representation**.

After those adjustments, since most of the machine learning models also require the target variable to be numeric, we applied **label encoding to the target variable**. In this way, each unique category in the target variable was mapped to an integer value.

Target encoding was then done for variables of object type. With this technique, each category of a categorical feature is replaced by the mean of the target variable within that particular category. This allows the model to learn well the relationships between the categorical feature and the target, which can improve the performance of models, especially with high cardinality categorical variables *(Pargent et al., 2022)*.

Finally, we scaled all the features with **MinMax Scaler**. Scaling is a very important step to make sure features contribute equally to the model performance when features are in different units or magnitudes. Because the scaler is **fitted on the train data** alone, the model never gets a prior knowledge about the test set, therefore keeping the validity of the evaluation process. Once fitted, the same scaling parameters were applied to the training and testing datasets.

Multiclass Classification

Feature Selection

The first step of our feature selection process was to **exclude all date variables**, since our objective was to build a model for predicting future claims and the date variables values can be specific to the training dataset and not applicable to new claims. In this way, we made sure that the model was generalizable and not trained to time-specific information.

After excluding the date variables, we applied a few methods for feature selection *(Bouchlaghem et al., 2022)*.

First, we plotted a **Spearman Correlation Matrix** *(figure 10)* to detect high correlations between features, which could introduce redundancy in the model. With this first check, we identified high correlations between *Birth Year* and *Age at Injury, Average Weekly Wage and Above County Wage Average*, *IME-4 Count* and *Multiple Form Submissions*, *Number of Dependents* and *Wage/Dependents Ratio*, and *Time to Assembly* and *Time to C-2*.

Second, we made use of the **Random Forest Classifier Feature Importances** *(figure 11)* to determine the most important features to make accurate predictions. The choice of this method was related to its good performance on complex datasets and its accountability for interactions between features. After evaluating the results, we decided to apply a 2% threshold to filter the less informative features, meaning that any feature with importance greater than 2% is considered relevant, since it represented the midpoint value.

Next, we implemented a **Mutual Information analysis** (figure 12) to measure how much each feature reduced uncertainty in predicting the target variable. We used this method because it captures both linear and non-linear relationships and works with both numerical and categorical features. With the same reasoning as in RFCFI, we applied a 2% threshold to filter the features.

The final method we used was **Lasso Regression** (figure 13). This method's results are very easy to interpret: it shrinks the coefficients of the less important features to zero.

To conclude the feature selection, we used a **voting system** based on the results of the three main methods: RFCFI, MI, and LR. This process can be explored in detail in *Figure 14*. A feature was retained if it was selected by at least two of these methods, while features chosen by only one method were discarded. Additionally, we used the results from the Spearman Correlation analysis to make sure that only one feature from each pair of highly correlated variables was kept.

After this analysis, 16 features were kept, meaning that we were able to reduce the number of variables by almost 50%.

Model Comparison and Selection

Before diving in all the models that we tested, it is important to talk about the model validation method that we choose. That method was the **holdout method** *(Kumar, 2023)* and the reasons for that choice are very practical: given the large size of our train dataset, and the fact that we opted not to apply any method to reduce it, the holdout method is an efficient choice as it splits the data into training and validation sets without the additional computational complexity associated with cross-validation. This decision was not an easy one, since we recognize that cross-validation can provide more robust results, but its huge computational demands made it inefficient to test all the models that we wanted to test in our large dataset.

Regarding the evaluation metric, we used **F1 Weighted** because it balances precision and recall, reflects the actual class distribution, and avoids over-representing minor classes that may have a reduced impact on the overall model performance. This is important in our context, since we have a multiclass classification problem with highly imbalanced classes (showed below). We believe that this

metric is fair and accurate when it comes to evaluating and comparing model's performance, especially in cases like ours where the train datasets are imbalanced.

```
Claim Injury Type
2. NON-COMP          232862
4. TEMPORARY         118806
3. MED ONLY           55125
5. PPD SCH LOSS       38624
1. CANCELLED           9981
6. PPD NSL             3369
8. DEATH                376
7. PTD                   77
Name: count, dtype: int64
```

Figure 1: Distribution of our target variable

Moving to the actual model comparison and selection, we tested seven different models.

We started with **RandomForestClassifier**, a very popular ensemble model that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting (*RandomForestClassifier*, n.d.). Next, we tried the **DecisionTreeClassifier**. These two first models tend to overfit, so we controlled them with **parameters** like max_depth and min_samples_split.

The third model was the **HistogramGradientBoostingClassifier**, another ensemble method. HGBT uses gradient boosting to iteratively improve the model's performance by fitting each tree to the negative gradient of the loss function with respect to the predicted value. When training a multiclass classification model with a large number of possible classes, which can be our case, HGBDT fits internally one-tree per class at each boosting iteration *(Comparing Random Forests and Histogram Gradient Boosting Models, n.d.)*.

The **Multilayer Perceptron Classifier (MLP)**, a neural network model, was also tested, with careful parameters choices. We used 'adam' solver, as it works better on large datasets, and 'relu' activation, with 3 hidden layers each with twice the number of features. This number came from several different other attempts. We also used a learning_rate_init of 0.0025, a number chosen to balance good learning with computational complexity, and a tolerance for optimization (tol) of 1e-6. We set early stopping to True to end the training when validation score (it automatically sets aside 10% of training data as validation) is no longer improving *(MLPClassifier, n.d.)*.

The last three models tested were all ensemble models. Given our imbalanced training set, we tried **AdaBoostClassifier**, as it is known for focusing on more difficult cases, which could in our case be the minority classes *(AdaBoostClassifier, n.d.)*. This was done using a DecisionTreeClassifier as the base model. Then, we tried a **VotingClassifier** with RandomForestClassifier, DecisionTreeClassifier, HistGradientBoostingClassifier, and MLPClassifier as estimators. Finally, we implemented a **StackingClassifier** with RandomForestClassifier and HistGradientBoostingClassifier as estimators, and MLPClassifier as the final estimator.

To compare all the models, we generated a table with train F1 scores, validation F1 scores, and the time taken, in seconds, to also consider computational efficiency:

	Model	Train F1 Score	Validation F1 Score	Time Taken (s)
0	Random Forest	0.667470	0.667113	101.547307
1	Decision Tree	0.666550	0.666337	2.378379
2	Hist Gradient Boosting	0.703243	0.702366	6.780705
3	MLP Classifier	0.706393	0.705592	425.392422
4	Ada Boost Classifier	0.595093	0.582688	145.250923
5	Voting Classifier	0.686629	0.686242	483.148938
6	Stacking Classifier	0.717741	0.715405	448.839491

Table 1: Models Comparison

One could say that **HistGradientBoosting** stands out as a very fast model that achieves very similar results comparing with other ensemble models. However, we ended up taking **StackingClassifier** to our model improvement process, as it has a much larger room for improvement, and also includes HistGradientBoosting in its prediction method, as an estimator.

Model Improvement

In this phase, with the goal of improving the StackingClassifier performance, we tried to improve the performance of each of its estimators individually. To do that, we made use of **Grid Search Cross Validation**.

Starting with **RandomForestClassifier**, three different n_estimators (40, 50, and 60) were tested, along with three different max_depth (9, 13, and 15). The results showed that the best combination was n_estimators=60 and max_depth=15.

For **HistGradientBoostingClassifier**, we also managed max_depth (9, 13, and 15), and three options of max_iter (500, 1000, 2000). The best combination was max_depth=15 and max_iter=500.

Lastly, for **MLPClassifier**, since in the previous section we had already tried different parameters, we only checked if decreasing the learning_rate_init would improve the model. The results showed that with the initial learning rate that we had set (0.0025) the model performed better than when we decreased it to 0.001.

After applying the changes to the **StackingClassifier**, we got to a **final model** that achieved a validation **F1 Weighted score** of approximately **0.7255**.

Extra Preprocessing

Before applying our model to the test dataset, we noticed that the feature **Agreement Reached**, a meaningful one, was missing. To address this problem, we built a separate model to predict this feature using the available features.

To do so, we had to fall back on the preprocessed data before encoding and scaling, giving that we had encoded categorical features using target encoding based on Claim Injury Type, which could have introduced bias and misled the Agreement Reached prediction model. For that reason, we had to perform target encoding, this time based on the target variable Agreement Reached, and scaling again.

After that, we used a very similar **feature selection** approach to the one we used in the main model, with only one adjustment: the threshold for the Random Forest Classifier (RFC) was set to 0.5% instead of 2%, as very few features achieved the 2% value.

For model comparison and selection, we evaluated four models: **Random Forest**, **Histogram Gradient Boosting**, **MLP Classifier**, and **Stacking Classifier**. After comparing their overall scores, the **Stacking Classifier** was selected and applied.

With this done, we were finally able to apply our main model.

Open-Ended Section

For this final section of our project, we had a very clear goal: to develop an **easy-to-operate** and **user-friendly platform** for WCB staff to predict the Claim Injury Type of as many workers as they need. To do this, we made use of **Streamlit**, a very simple and fast tool *(Streamlit, n.d.)*.

The interface we created has two columns with different **features to be filled** (the features we used to train our model), and a ***Predict*** bottom at the end of the page *(Figure 15)*. Once the inputs are submitted, the app processes the values using the trained machine learning model to generate immediate predictions. This functionality highlights the model's adaptability to new data and its potential role in supporting decision-making.

Besides providing the prediction, the app also displays a **confidence score** derived from the ***predict_proba*** method of the **StackingClassifier**. This score represents the highest probability assigned to a class, indicating the model's confidence in its prediction. In addition, for transparency, it displays a graph with the **feature importance scores**, that come from the RandomForestClassifier estimator, indicating which data points had more influence in the prediction (Figure 16).

This interface transforms the model's predictive capabilities into actionable insights, making it accessible to non-technical users. It bridges the gap between technical development and **practical application**, demonstrating how decision-makers can use machine learning outputs to assess claims effectively and interactively.

Conclusion

As we bring our project to a close, we can proudly reflect on how we have successfully met our primary objectives: we were able to find a **good solution to the inefficiencies and challenges faced by WCB staff**. Apart from developing a predictive model capable of determining the Claim Injury Type within seconds, we have created a tool that is not only efficient but also **user-friendly**, making it accessible even to individuals with zero technical knowledge.

To accomplish our goal, we went by some key steps. The preprocessing stage was particularly intensive, starting with the resolution of basic issues such as data inconsistencies or duplicated rows, and extending to handling outliers, treating missing data, and correcting variable data types. Another critical task was to structure a clear feature selection strategy, and by doing so we were able to reduce the number of features by nearly 50% without compromising the quality of our model. Lastly, in the modeling phase, we explored a wide range of different machine learning algorithms, including traditional ones such as DecisionTreeClassifier and the more complex ones such as neural networks and ensemble techniques. After testing all the models, we got to the more promising one – StackingClassifier – but we did not stop there. We recognize the potential for further improvement, and we tried to do so making us of Grid Search Cross Validation. We achieved **a final model with a F1 Weighted score of 0.7255**, exceeding our initial expectations, but we are still aware that there are some points that could be improved.

Throughout the project, we faced some **challenges**. The **large size of the dataset** presented some computational difficulties, requiring careful management of resources and strategic decisions. For example, we would have liked to implement cross validation techniques for model evaluation, but with the large number of models that we wanted to test, it was not feasible. Another big challenge was the **imbalance in our target variable**, which we believe that limited our model performance. We have explored preliminary solutions, but we recognize that further work could include adopting advanced techniques to balance the target variable. For instance, **undersampling** the majority classes could simultaneously address dataset size issues, while **oversampling** minority classes using methods like SMOTE, or even more sophisticated techniques, could guarantee a more equal representation of all the eight classes.

While we are satisfied with the progress we have made, we view this work as a **foundation** upon which future studies can be built, opening the door to even greater impact and efficiency.

Annexes

```
RangeIndex: 593471 entries, 0 to 593470
Data columns (total 33 columns):
 #   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
dtypes: float64(11), int64(1), object(21)
memory usage: 149.4+ MB
```

Figure 1

BOXPLOTS BEFORE OUTLIER TREATMENT

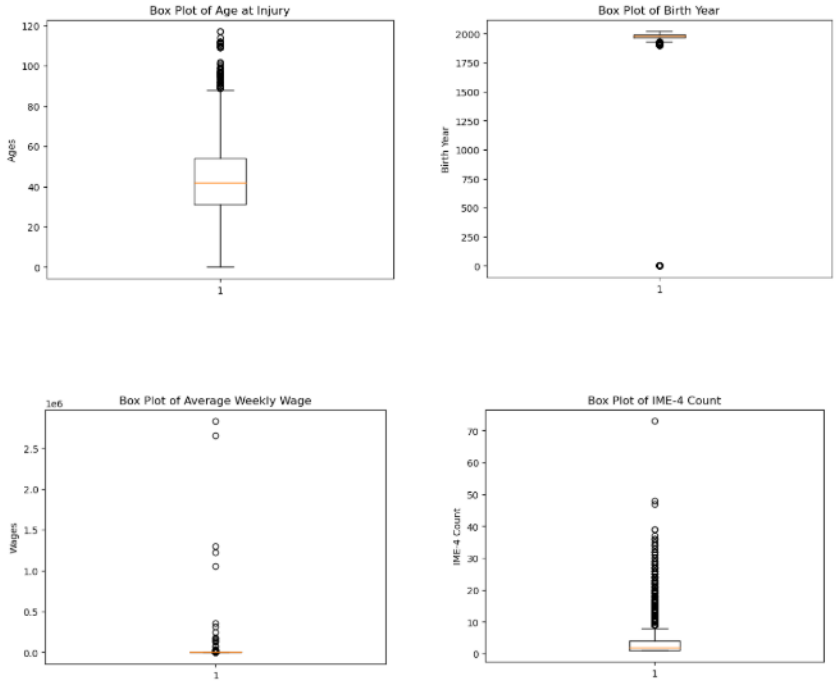

Figure 2

BOXPLOTS AFTER OUTLIER TREATMENT

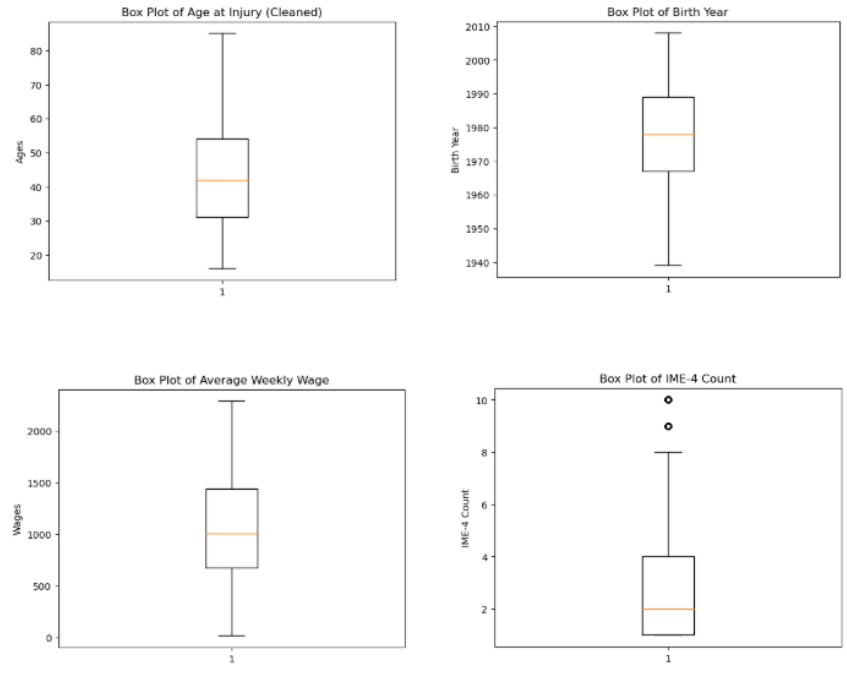

Figure 3

Claim Identifier	Accident Date	Injury Day of Week
5785935	2021-08-05	Thursday
5980545	2022-04-05	Tuesday
5552635	2020-09-01	Tuesday
5758039	2021-06-10	Thursday
5951382	2022-01-26	Wednesday

Figure 4

Claim Identifier	Accident Date	Assembly Date	Time to Assembly
5785935	2021-08-05	2021-08-10	5
5980545	2022-04-05	2022-05-02	27
5552635	2020-09-01	2020-09-04	3
5758039	2021-06-10	2021-07-02	22
5951382	2022-01-26	2022-03-25	58

Figure 5

Claim Identifier	Accident Date	C-2 Date	Time to C-2
5785935	2021-08-05	2021-08-10	5
5980545	2022-04-05	2022-04-30	25
5552635	2020-09-01	2020-09-04	3
5758039	2021-06-10	2021-07-02	22
5951382	2022-01-26	2022-03-30	63

Figure 6

Claim Identifier	Average Weekly Wage	Number of Dependents	Wage/Dependents Ratio
5785935	744.060000	6.0	124.01
5980545	1157.330000	2.0	578.66
5552635	1047.017294	3.0	349.01
5758039	824.590881	1.0	824.59
5951382	250.000000	4.0	62.50

Figure 7

Claim Identifier	IME-4 Count	Multiple Form Submissions
5785935	2.0	1
5980545	0.0	0
5552635	0.0	0
5758039	0.0	0
5951382	0.0	0

Figure 8

Claim Identifier	County of Injury	Average Weekly Wage	Above County Wage Average
5785935	QUEENS	744.060000	0
5980545	BRONX	1157.330000	1
5552635	KINGS	1047.017294	0
5758039	KINGS	824.590881	0
5951382	NASSAU	250.000000	0

Figure 9

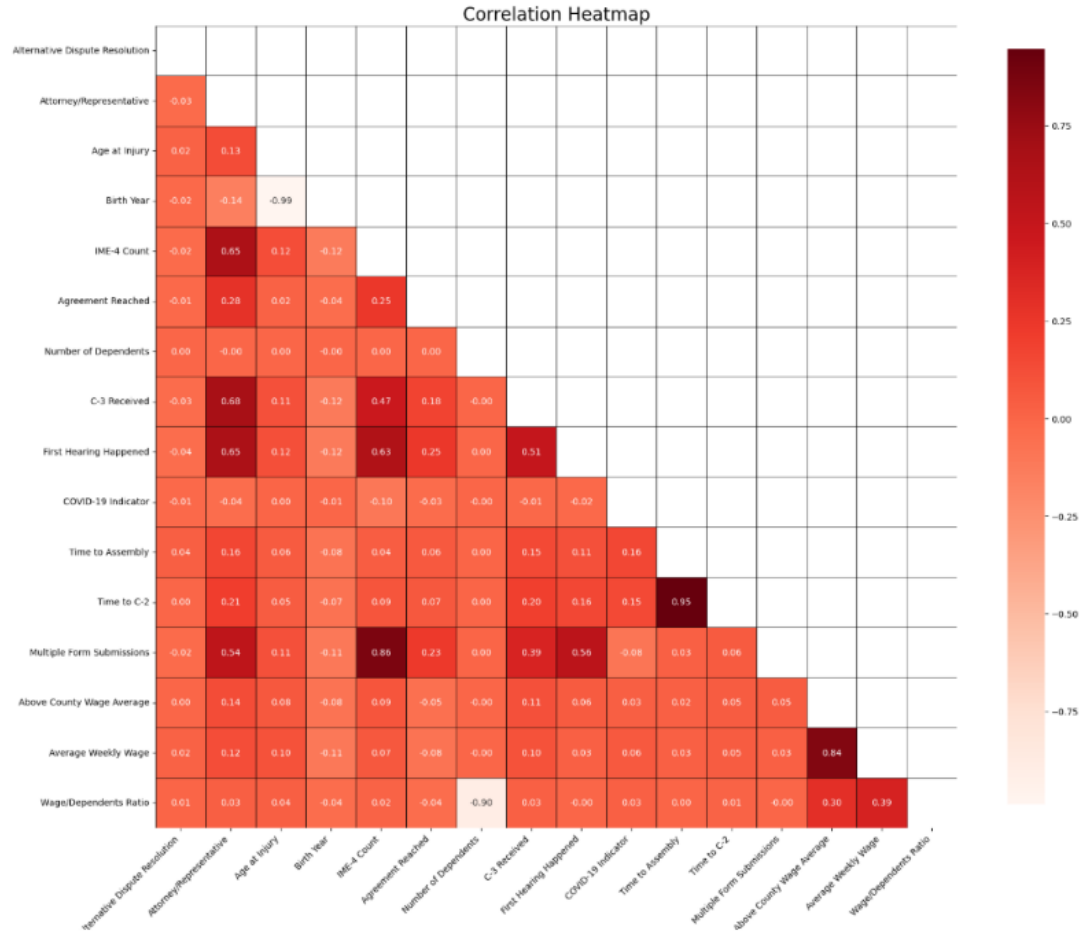

Figure 10

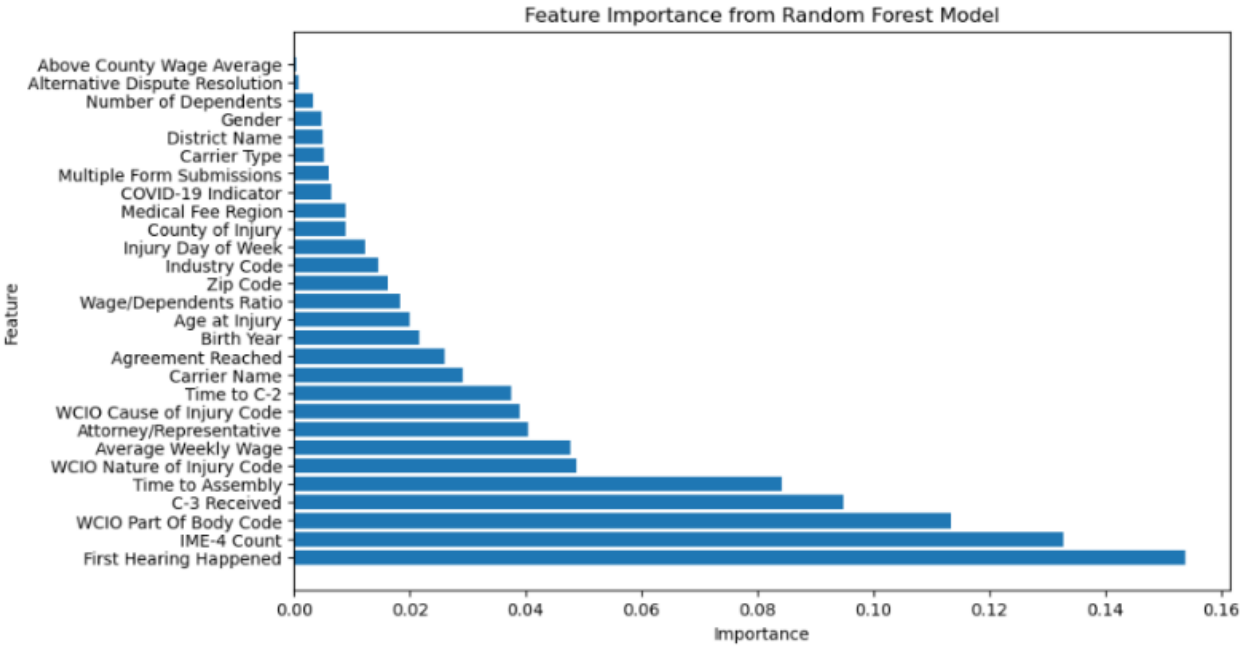

Figure 11

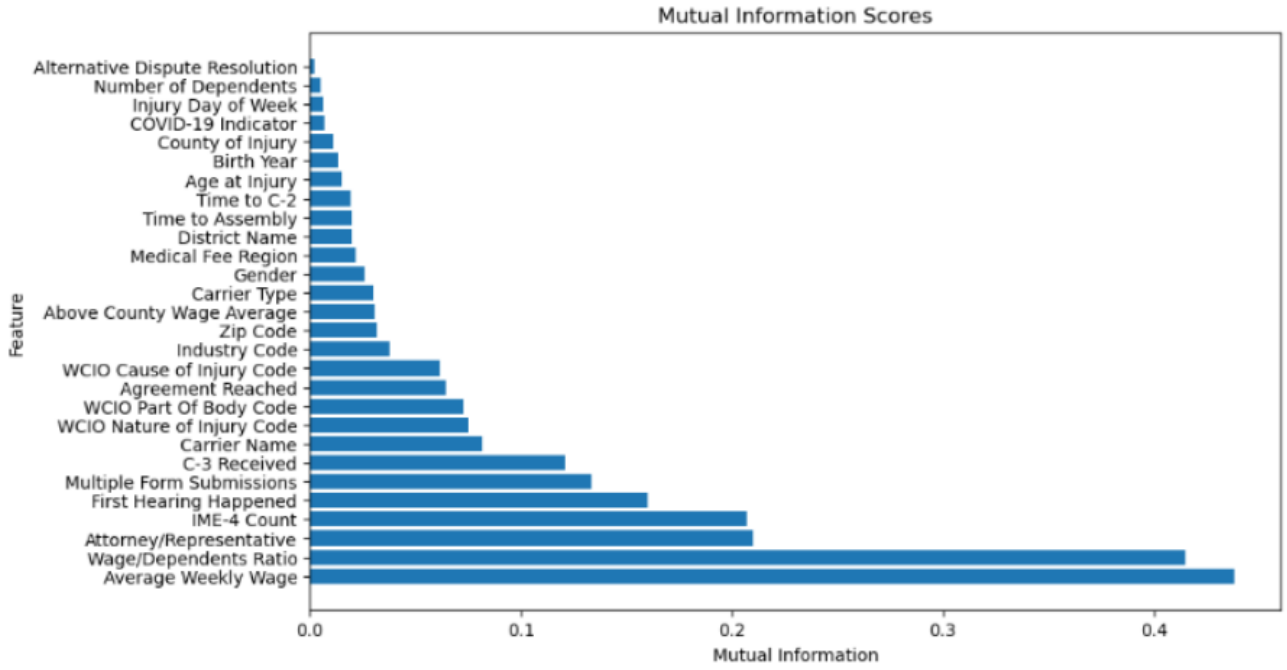

Figure 12

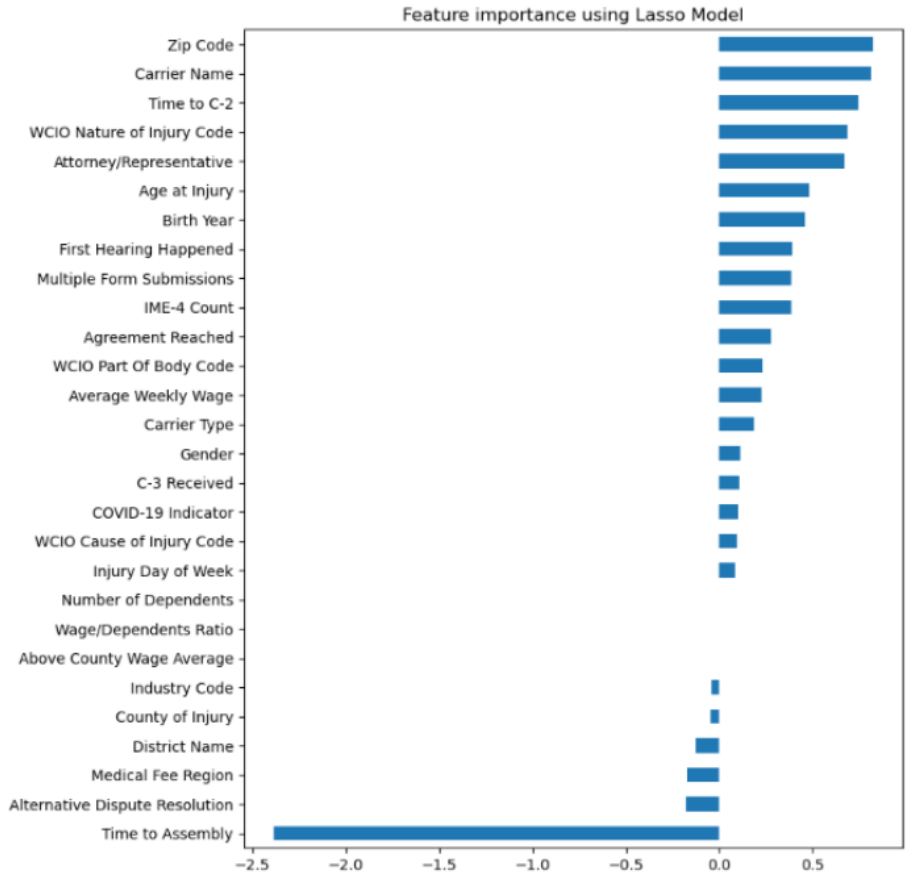

Figure 13

Predictor	Spearman	RandomForestClassifier Feature Importance	Mutual Information	LASSO Regression	What to do? (One possible way to "solve")
Age at Injury	High correlated with Birth Year	Discard	Discard	Keep	Discard
Alternative Dispute Resolution	Keep	Discard	Discard	Keep	Discard
Attorney/Representative	Keep?	Keep	Keep	Keep	Include in the model
Average Weekly Wage	High correlated with Above County Wage Average	Keep	Keep	Keep	Include in the model
Birth Year	High correlated with Birth Year	Keep	Discard	Keep	Include in the model
Carrier Name	NA	Keep	Keep	Keep	Include in the model
Carrier Type	NA	Discard	Keep	Keep	Include in the model
County of Injury	NA	Discard	Discard	Keep	Discard
COVID-19 Indicator	Keep	Discard	Discard	Keep	Discard
District Name	NA	Discard	Discard	Keep	Discard
Gender	NA	Discard	Keep	Keep	Discard
IME-4 Count	High correlated with Multiple Form Submissions	Keep	Keep	Keep	Include in the model
Industry Code	NA	Discard	Keep	Keep	Include in the model
Medical Fee Region	NA	Discard	Keep	Keep	Include in the model
WCIO Cause of Injury Code	NA	Keep	Keep	Keep	Include in the model
WCIO Nature of Injury Code	NA	Keep	Keep	Keep	Include in the model
WCIO Part Of Body Code	NA	Keep	Keep	Keep	Include in the model
Zip Code	NA	Discard	Keep	Keep	Include in the model
Agreement Reached	Keep	Keep	Keep	Keep	Include in the model
Number of Dependents	High correlated with Wage/Dependents Ratio	Discard	Discard	Discard	Discard
C-3 Received	Keep	Keep	Keep	Keep	Include in the model
First Hearing Happened	Keep	Keep	Keep	Keep	Include in the model
Injury Day of Week	NA	Discard	Discard	Keep	Discard
Time to Assembly	High correlated with Time to C-2	Keep	Keep	Keep	Include in the model
Time to C-2	High correlated with Time to Assembly	Keep	Discard	Keep	Discard (already kept Time to Assembly)
Wage/Dependents Ratio	High correlated with Number of Dependents	Discard	Keep	Discard	Discard
Multiple Form Submissions	High correlated with IME-4 Count	Discard	Keep	Keep	Discard (already kept IME-4 Count)
Above County Wage Average	High correlated with Average Weekly Wage	Discard	Keep	Keep	Discard (already kept Average Weekly Wage)

Figure 14

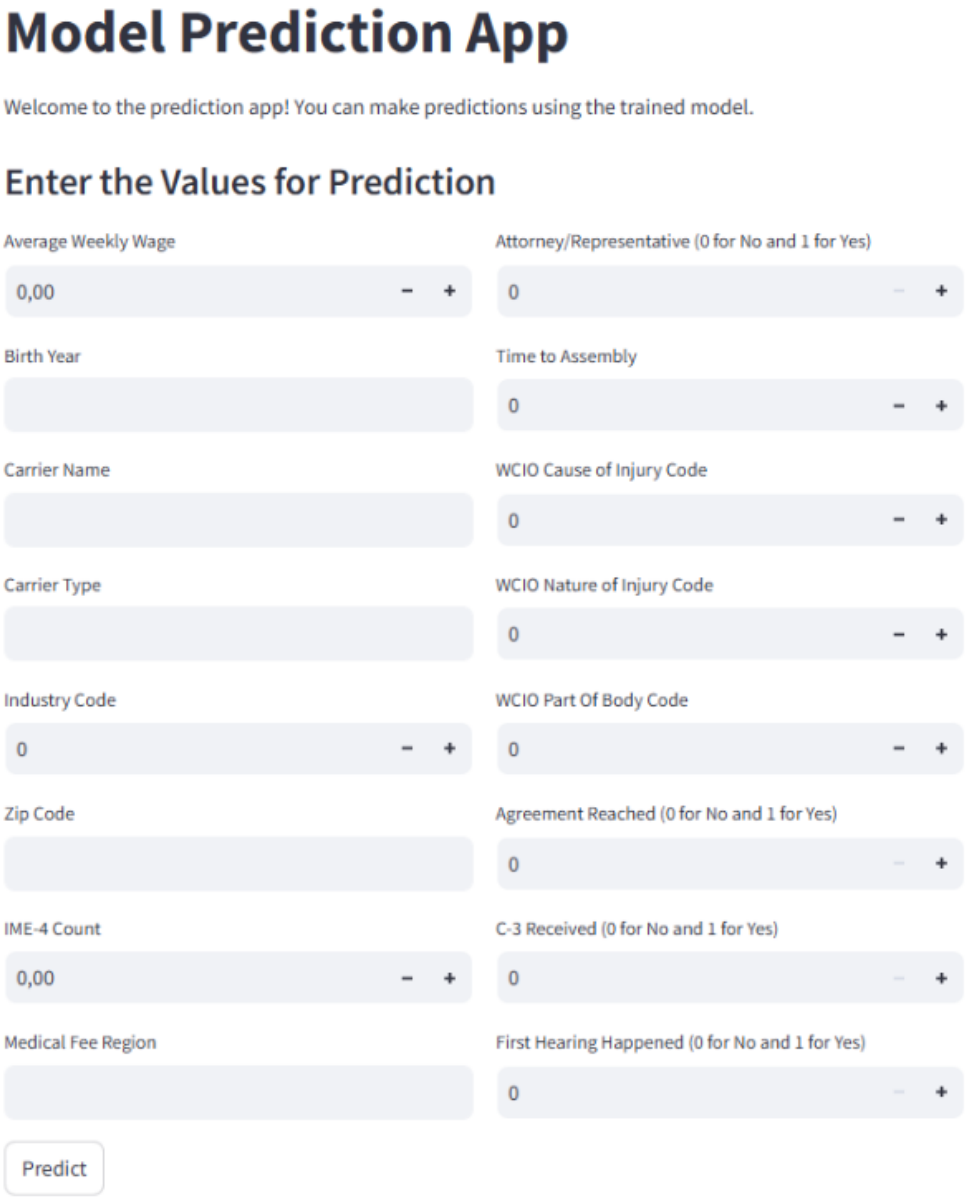
Model Prediction App
Welcome to the prediction app! You can make predictions using the trained model.
Enter the Values for Prediction
Average Weekly Wage
0,00
Attorney/Representative (0 for No and 1 for Yes)
0
Birth Year
Time to Assembly
0
Carrier Name
WCIO Cause of Injury Code
0
Carrier Type
WCIO Nature of Injury Code
0
Industry Code
0
WCIO Part Of Body Code
0
Zip Code
Agreement Reached (0 for No and 1 for Yes)
0
IME-4 Count
0,00
C-3 Received (0 for No and 1 for Yes)
0
Medical Fee Region
First Hearing Happened (0 for No and 1 for Yes)
0
Predict

Figure 15

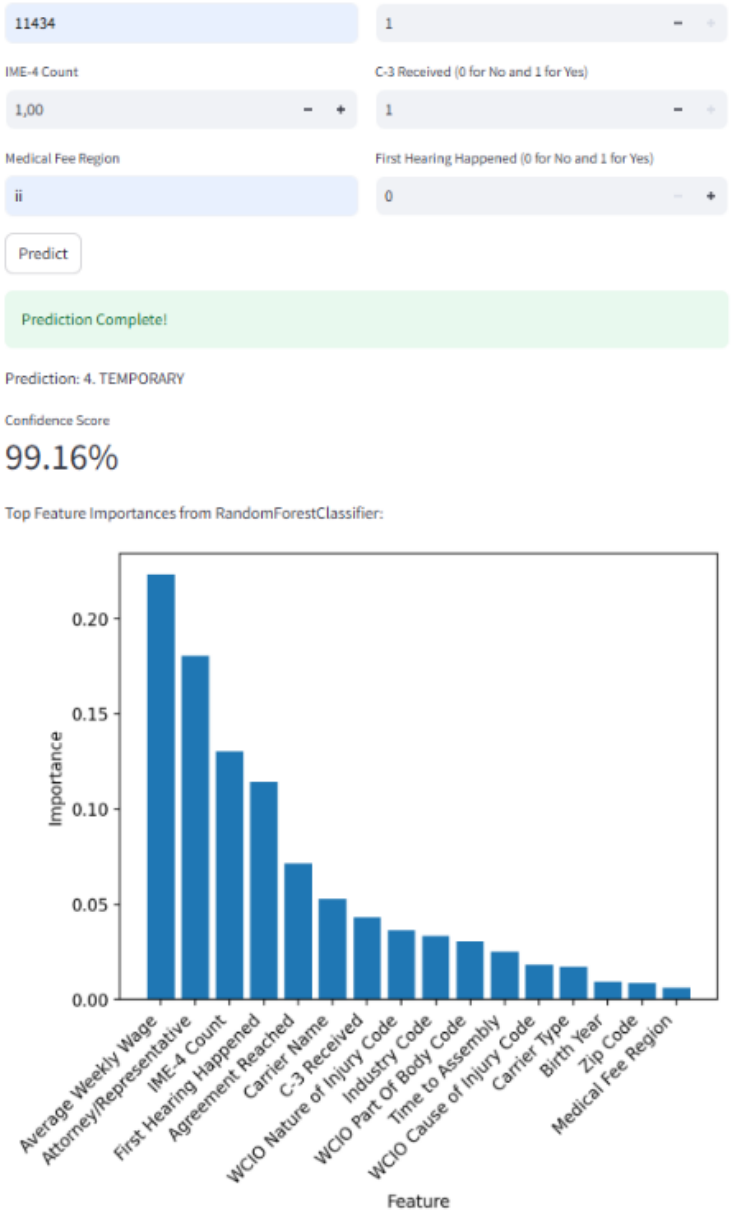
11434
1
IME-4 Count
1,00
C-3 Received (0 for No and 1 for Yes)
1
Medical Fee Region
ii
First Hearing Happened (0 for No and 1 for Yes)
0
Predict
Prediction Complete!
Prediction: 4. TEMPORARY
Confidence Score
99.16%
Top Feature Importances from RandomForestClassifier:
Importance
0.20
0.15
0.10
0.05
0.00
Average Weekly Wage
Attorney/Representative
IME-4 Count
First Hearing Happened
Agreement Reached
Carrier Name
C-3 Received
WCIO Nature of Injury Code
Industry Code
WCIO Part Of Body Code
Time to Assembly
WCIO Cause of Injury Code
Carrier Type
Birth Year
Zip Code
Medical Fee Region
Feature

Figure 16

References

- AdaBoostClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
- Bouchlaghem, Y., Akhiat, Y., & Amjad, S. (2022). Feature Selection: a review and comparative study. E3S Web of Conferences, 351, 01046. https://doi.org/10.1051/e3sconf/202235101046
- Comparing Random Forests and Histogram Gradient Boosting models. (n.d.). Scikit-learn. https://scikit-learn.org/1.5/auto_examples/ensemble/plot_forest_hist_grad_boosting_comparison.html#sphx-glr-auto-examples-ensemble-plot-forest-hist-grad-boosting-comparison-py
- Gelman, A., & Hill, J. (2006). Missing-data imputation. In Cambridge University Press eBooks (pp. 529–544). https://doi.org/10.1017/cbo9780511790942.031
- Handling missing data in data Pre-Processing. (n.d.). Bigelectrons. https://bigelectrons.com/post/mlandai/handling-missing-data/
- Kakhki, F. D., Freeman, S. A., & Mosher, G. A. (2019). Evaluating machine learning performance in predicting injury severity in agribusiness industries. Safety Science, 117, 257–262. https://doi.org/10.1016/j.ssci.2019.04.026
- Khairuddin, M. Z. F., Hui, P. L., Hasikin, K., Razak, N. a. A., Lai, K. W., Saudi, A. S. M., & Ibrahim, S. S. (2022). Occupational injury risk Mitigation: machine learning approach and feature optimization for smart workplace surveillance. International Journal of Environmental Research and Public Health, 19(21), 13962. https://doi.org/10.3390/ijerph192113962
- Kumar, A. (2023, May 22). Hold-out method for training machine learning models - Analytics Yogi. Analytics Yogi. https://vitalflux.com/hold-out-method-for-training-machine-learning-model/
- Liu, H. (2018b). Feature engineering for machine learning and data analytics. In CRC Press eBooks. https://doi.org/10.1201/9781315181080
- MLPClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
- Pargent, F., Pfisterer, F., Thomas, J., & Bischl, B. (2022). Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. Computational Statistics, 37(5), 2671–2692. https://doi.org/10.1007/s00180-022-01207-6
- RandomForestClassifier. (n.d.). Scikit-learn. https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- Sarkar, S., Vinay, S., Raj, R., Maiti, J., & Mitra, P. (2018). Application of optimized machine learning techniques for prediction of occupational accidents. Computers & Operations Research, 106, 210–224. https://doi.org/10.1016/j.cor.2018.02.021