Modeling for Auto Insurance Claims

A Reliable and Actionable Predictive Model for Auto Claims Identification in Insurance

Prepared by Team Facturial

2024 ASNA Hackathon

Team Members:

Eric Li
Kevin Jiang
Ki Yan Chan
Weidong Zhang

Table of Contents

Introduction

Trained minds can see where intuition falters.

Actuarial professionals in the insurance industry are such “trained” personnel, equipped with the skills to draw meaningful conclusions from vast and complex data sets. Such competency is particularly salient in the process of underwriting, where, historically, actuaries process relevant data clinically and decide on pricing for individual clients. (Batty, n.d.) Therefore, it is essential for actuaries to understand the distinctions between clinical and actuarial judgment in underwriting, with the latter being a data-driven approach that often proves more reliable. While clinical judgment has long been a tradition in actuarial practice, research shows that actuaries gain substantial benefits from using statistical modeling and data mining to arrive at more rigorous, “actuarial” conclusions (Batty, n.d.).

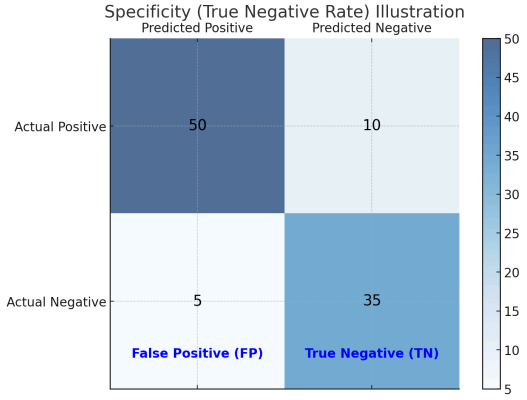
This report aims to present a practical model for actuarial firms to apply in auto claims modeling. It begins with a brief discussion of the challenges in the auto claims industry and the metrics considered in our model. Next, we outline our methodology for designing the model’s functions, followed by implementation details and performance evaluations across various metrics. Subsequent sections forecast the model’s potential business impact and clarify the steps necessary to achieve specific business outcomes. Finally, we address the model’s current limitations, and suggest potential extensions based on additional metrics.

Business Context and Objectives

Businesses wish to analyze the qualities and characteristics of the relationships and dependencies between gathered client data and client claim size to create actionable, meaningful business decisions. This principle equally applies to the insurance industry, where every insurer strives to find the universal link between a client's history and the likelihood or size of a claim. (Promutuel Insurance, n.d.)

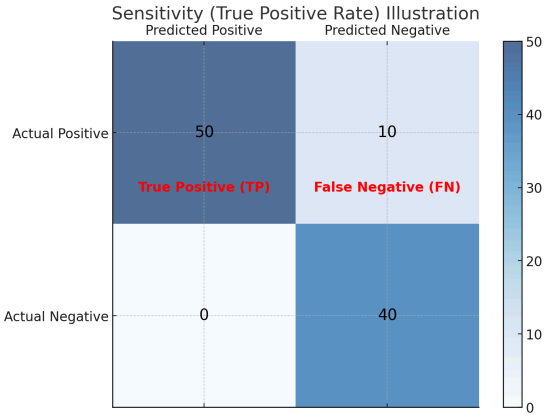
An improved claims prediction model can assist actuaries in tackling persistent modeling challenges by focusing on two key metrics: accuracy and efficiency. For a claims model that yields binary outcomes, accuracy is often evaluated through *specificity* and *sensitivity*. Specificity measures the model's ability to identify true negatives (accurately predicting no claims) (*Sensitivity ,Specificity, and Accuracy: Understanding Model Performance*, 2024):

Figure 1: Visual illustrations of false positive and true negative in a confusion matrix



Sensitivity, on the other hand, reflects its effectiveness in identifying true positives (accurately predicting claims) (*Sensitivity, Specificity, and Accuracy: Understanding Model Performance*, 2024):

Figure 2: Visual illustrations of true positive and false negative in a confusion matrix



Both metrics have substantial implications for business operations. For instance, a lack of specificity—failing to identify true negatives accurately—or sensitivity—failing to identify true positives—could lead to skewed claim estimates. This may result in either underestimating potential costs and increasing risk or overestimating costs and adopting an overly cautious investment strategy. Therefore, a well-designed model seeks a delicate balance between specificity and sensitivity (Walach et al., 2021), minimizing the risk of biased predictions and supporting more informed decision-making in underwriting.

Efficiency, on the other hand, is vital to an insurance company's operational performance. In this, the currently used Generalized Linear Models—a model traditionally favored by the insurance industry—present notable challenges. First, underwriting is costly. In the life insurance sector, for instance, an insurer typically spends approximately one month and several hundred dollars underwriting each applicant (Batty, n.d.). These expenses, coupled with a shrinking pool of underwriting professionals, make the current process increasingly unsustainable. Furthermore, the lengthy process in obtaining insurance coverage can deter potential clients, creating frustration due to wait times (Batty, n.d.). Improving efficiency in decision-making could make purchasing insurance from your company more attractive. Additionally, a robust predictive model could help identify patterns among highly qualified clients, aiding in targeted marketing—an essential cost-saving strategy for actuarial businesses, as marketing is a significant expense in the life insurance industry (Tiemann, 2019).

Now, imagine a model that excels in both metrics, enhancing prediction accuracy while streamlining operations. Although the traditional GLM model is intuitive and easy to use, it falls short of meeting these growing demands (Society of Actuaries, 2020). A powerful modern tool aligned with this dual objective, offering an ideal solution for building the models needed, is machine learning. (Hearst, n.d.)

Model Explanation

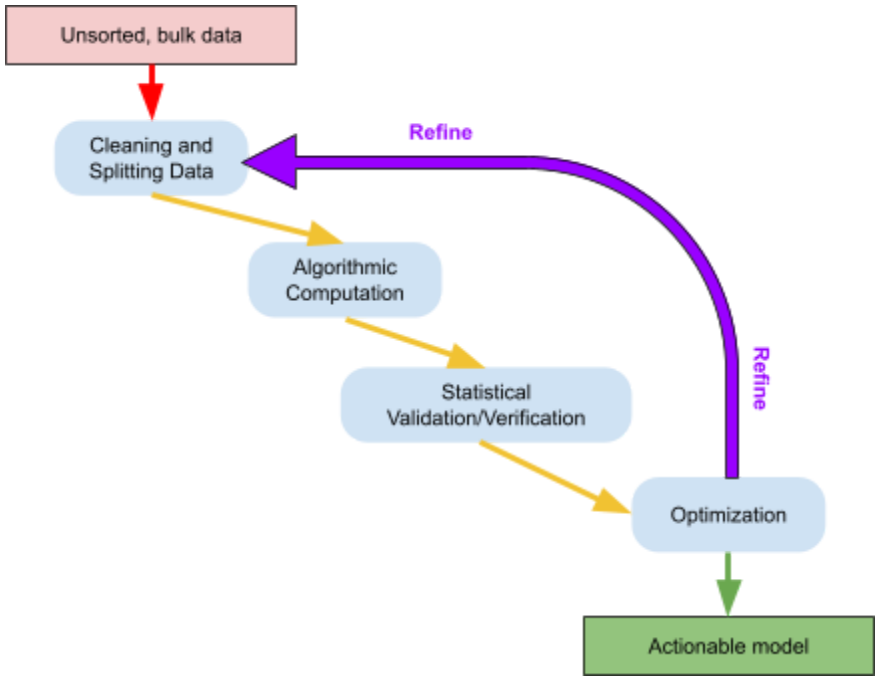
Machine learning is incredibly powerful in its ability to harness the power of electronic computers to identify these sought-after relationships being inputs and outputs (Brown, 2021). In the specific context, our inputs involve client information, and our outputs their claim size. In short, through machine learning we are able to develop a relationship model that summarizes the relationship between inputs and outputs.

Methodology

To the unacquainted, machine learning might seem like a black box. However, the model we used can be clearly explained in four key stages:

1. Data Preparation – Cleaning and splitting data into training and testing sets.
2. Algorithm Application – Using Decision Trees and Linear Regression.
3. Validation – Ensuring model accuracy through statistical checks.
4. Optimization – Fine-tuning for best predictive performance.

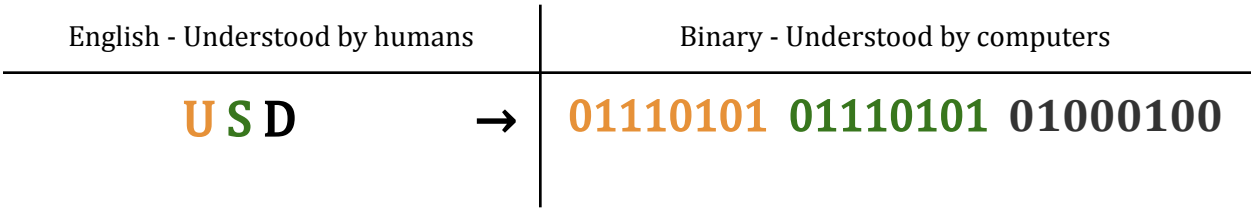
Figure 3: Visualize model development cycle for machine learning building



Cleaning and Splitting Data

Before processing and computing relationships, the data must be sorted and prepared to be fed into algorithmic calculations. Similar to setting variables in an equation, we translate human-understood language into a format understood by computers. This is achieved through various programmable functions, as shown in *Appendix A*, where numerical columns are "cleaned" for computation. The code used for this preparation is not directly relevant to the final conclusion and includes specific technical functions, whose explanation would only complicate this overview.

Figure 4: Overview of data cleaning, converting data into binary for computer interpretation



In summary, words are first converted into numbers, which are then formatted into two types of data:

1. **Continuous Data**

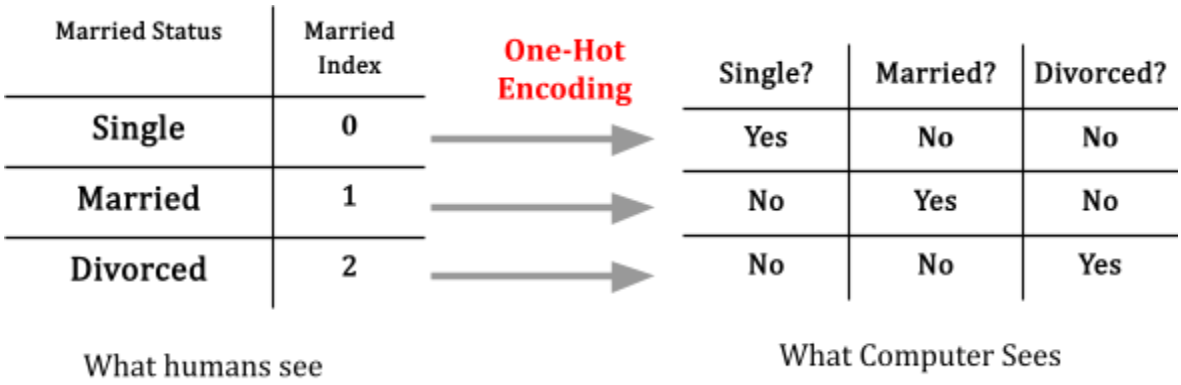
   Continuous data spans a spectrum or range of values (Zangre, 2024). An example from this report is Customer Lifetime Value, which cannot be easily sorted into specific, fixed categories. While categories could theoretically be created, each would encompass a range of values. Thus, treating it as continuous data is more straightforward. This approach was applied to the following categories:

   - Customer lifetime value
   - Income
   - Months since last claim
   - Months since policy inception

2. **Discrete Data**

   Discrete data is sorted into categories, typically because it describes qualities rather than numerical ranges (Zangre, 2024). For example, a person's marital status can be defined as "Single," "Married," or "Divorced" but cannot be expressed as "0.9 Married" or "50% Married" as with continuous data. Discrete data must be treated as distinct categories. One common method for categorizing data in computing is *one-hot encoding* (Jain, 2024). This technique converts each categorical value into binary states, indicating whether an item falls within a specific "qualitative" category. For example, marital status data would be converted into binary indicators for each status—"Single," "Married," or "Divorced." If a client is not in a category, their data is excluded from computations for that category. Code for this section is in *Appendix B*.

Figure 5: Overview of one-hot encoding, converting discrete data into binary indicators.



In summary, one-hot encoding is a common method to ensure that discrete data is treated differently from continuous data when applied in machine learning algorithms. The following data types were considered discrete and all underwent the same one-hot encoding technique:

Figure 6: Data types requiring one-hot encoding for machine learning being fed into a one-hot encoding function

```
Xs[i] = pd.get_dummies(X, columns=[
  'State',
  'Response',
  'Coverage_Index',
  'Education_Index',
  'Employment_Status_Index',
  'Marital_Status_Index',
  'Policy_Index',
  'Policy_Type_Index',
  'Sales_Channel_Index',
  'Vehicle_Size_Index',
  'Renew_Offer_Type'
])
```

Finally, some data was omitted from the machine learning model altogether. The rationale for removing certain types of data is detailed in the *Optimization* section.

Another key aspect of data processing is *splitting* (Gillis, n.d.), which involves dividing the data into training and testing sets. This allows the model to adjust and refine its predictions based on its performance on the testing set. While the importance of having both training and testing data will be explored further in the Statistical Validation section, the basic idea is that, like humans, the model needs "feedback" to confirm its accuracy. By splitting the data, the model can use this feedback to enhance its accuracy and reliability. The code for the data splitting process is provided in *Appendix H* and an explanation is provided in the Model Optimization of this paper.

Algorithmic Computation

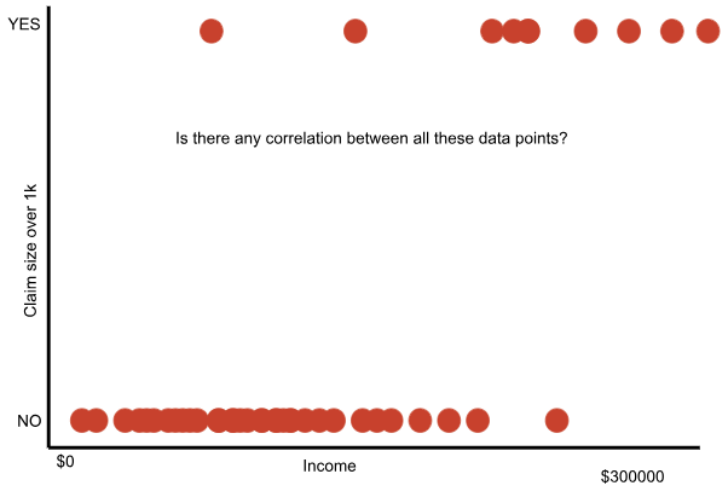
There are various mathematical methods to identify trends between quantities. In our report, however, we employ two straightforward methods—linear regression and decision trees—to determine relationships between the input data and our target outcome, claim size. The reasons why we selected these methods and how they operate are as follows:

1. **Linear Regression**

Linear regression is a statistical method used to determine the line of best fit for a set of continuous data between two variables. For instance, it could be applied to explore the relationship between a client's income and the likelihood of a large claim size. Linear regression excels at isolating the "signal" from the "noise" in data. A notable example of this is when Francis Galton analyzed 787 guesses of an ox's weight at a county fair (Yong, 2013). He discovered that while individual guesses varied, their errors balanced out, and the average guess came within one pound of the actual weight of the ox. This story demonstrates how regression models effectively refine predictions from diverse data points.

The model will then run computations to determine the line of best fit, aiming to identify if a linear relationship exists between the data points displayed on the graph. The computer applies an algorithm that analyzes the location of each point, calculating the best-fit line that predicts the general trend in the data. The exact mathematical formulas for linear regression are detailed in *Appendix C*, with a brief demonstration of its methodology and utility presented here.

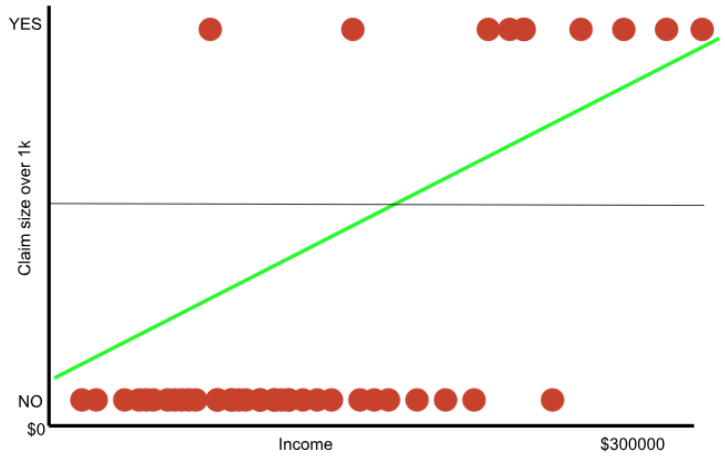
Figure 7: Correlation between data points of individual income and claim size over $1000



In the illustration below, the green line represents an approximation of this trend. This approximate line is generated through a method of gradient boosting (see *Appendix C*) XGBoost, a

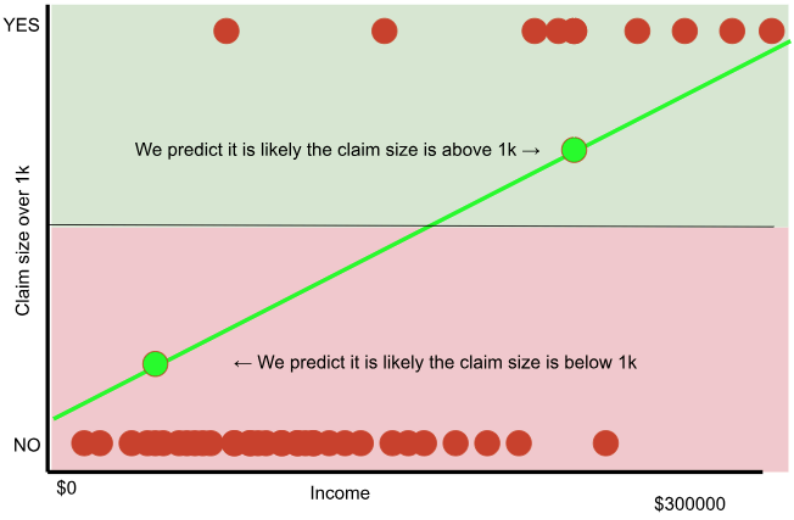
machine learning program, performs similar computations across various continuous data variables to enhance the model's accuracy in predicting outcomes.

Figure 8: Visual representation of machine learning applied to continuous data variables



Moving forward, the model will plot a client's data point on the line of best fit and, based on the region where the data point falls, predict the most likely outcome for the client.

Figure 9: Visual representation of how machine learning prediction functions



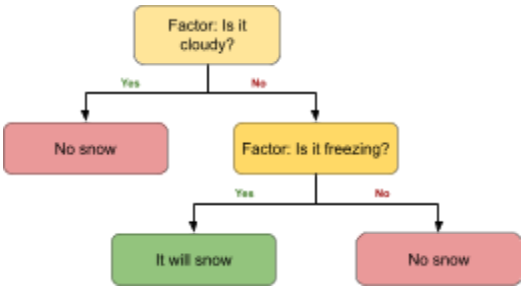
We chose linear regression because the foundation for insurance modeling is supported by findings of M. Fauzan and H. Murfi in their paper on Insurance Claim Prediction (Fauzan, n.d.), as well as D. Samson and H. Thomas in their study on linear models for Automobile Insurance Claims (Samson, n.d.). Both studies concluded that most continuous data factors show linear relationships with final claim sizes.

In short, linear regression is well-suited for handling continuous data in our model, and was integrated into our model through the use of XGBoost and SVM.

2. Decision Trees and Random Forest

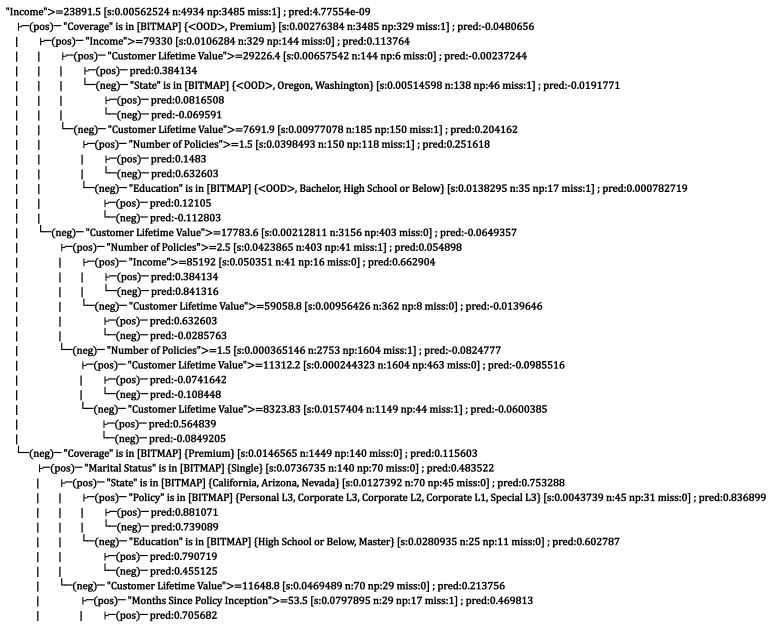
Decision trees are tree diagrams that use mathematical conditions to estimate the probability of specific outcomes. Different conditions and attributes lead to different paths, resulting in varied outcomes. A brief visual explanation of decision trees is shown below:

Figure 10: Overview of the process of decision tree models predictions



As demonstrated in the example above, different responses to each question lead to distinct outcomes and determine a unique path through the tree. In graph theory, each possible final outcome of a path is referred to as a *leaf*. Thus, depending on the state of each information category, the model arrives at a different leaf. Machine learning enhances this model by optimizing paths based on data. Below is a sample decision tree from *YDF*, illustrating a section of one the trees used in our model.

Figure 11: Illustrations of the partial decision-making process from one of the trees used in the stacking model.

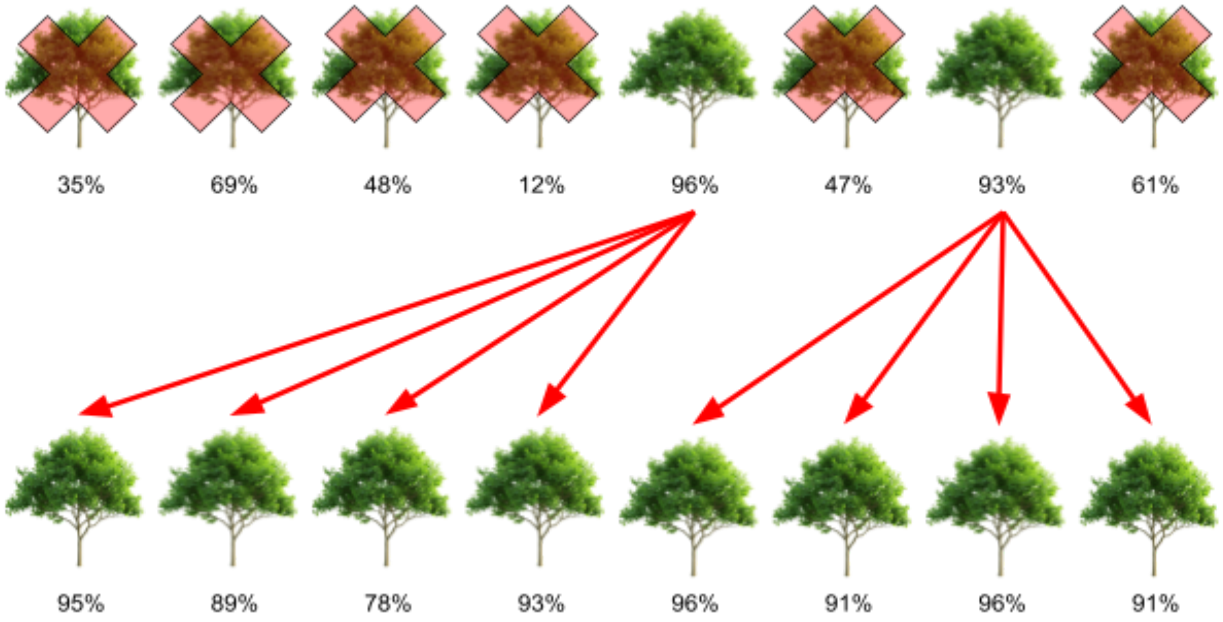


As shown in the figure above, decision trees are highly effective for modeling discrete data. Their Yes/No approach to determining outcomes aligns closely with the binary structure of discrete

data, such as that found in one-hot encoding. Consequently, the abundance of discrete data makes decision trees a natural choice for our machine learning model.

This raises the question: how are decision trees created? One method for building decision trees is called a random forest. First, we generate multiple trees using randomly selected attributes and outcomes. We then evaluate each tree's accuracy in predicting the model and "prune" those that perform poorly from the forest. Next, we create additional trees based on the "successful" trees, continuing this process until we achieve a consistent model of accurate trees.

Figure 12: Visual illustrations of random forest creating better decision trees



Thus, random forests are designed to develop effective decision trees. In this model, decision trees are implemented using the Python library *sklearn*, specifically through its Decision Tree module.

Stacking - The Combination of Multiple Machine Learning Models

As we have seen, linear regression is highly effective for handling continuous data, while decision trees are optimal for handling discrete data—both of which are present in our dataset. With both continuous and discrete data in our input, combining these machine learning models allows us to leverage the strengths of each, creating an even more robust predictive model (Alamir, 2021).

The combination of multiple models is referred to as *stacking*. Specifically, the models we stacked together successfully were:

1. XGBoost - a linear regression model that uses a novel and effective technique of gradient boosting to develop lines of best fit for continuous data (XGBoost, n.d.)
2. Decision Tree - a decision tree model from python *sklearn* that uses random forest to develop effective decision tree models for discrete data (scikit, n.d.)
3. YDF - a decision tree model from Google that uses random forest to develop effective decision tree models for discrete data (YDF, n.d.).
4. SVM - a linear regression model that uses higher-dimensional space linear regression techniques to accurately model data (Hearst, n.d.)

Furthermore, we see that when all four models are combined together, higher accuracy is obtained as seen in the figure below. The code for stacking models can be found in *Appendix D*.

Figure 13: Percentage accuracy from different approaches using a variety of models.

XGBoost	Decision Tree	YDF	SVM	Stacking Model: (combines all models mentioned above
90.5%	89.2%	90.1%	87.8%	**93.6%**

Model Optimization

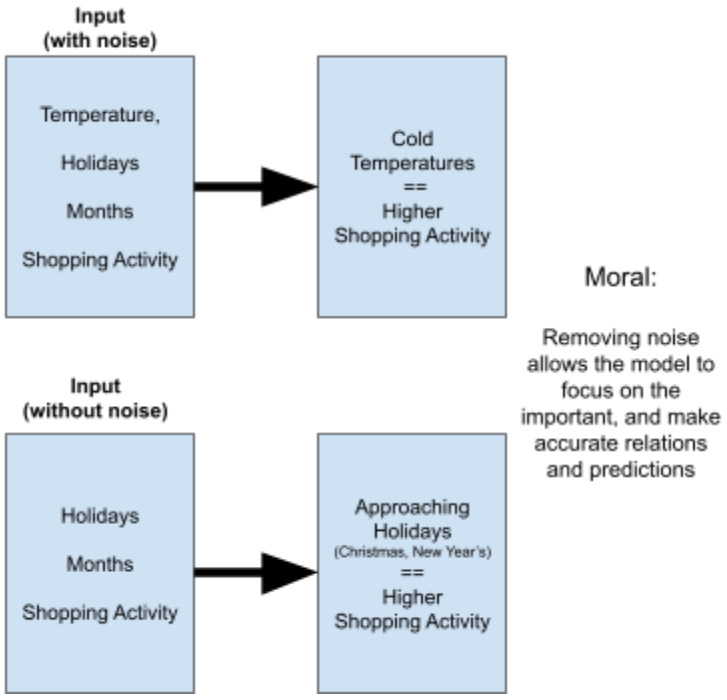
The final step to any machine model is optimization - refinement of the model to increase accuracy in its ability to detect relationships and predict outcomes. To perform this, three methods were considered:

1. Removing noise
2. Hyperparameter optimization
3. Undersampling and oversampling

Noise

Noise in machine learning refers to the idea of removing factors which have little, or no correlation with a final outcome. One *key assumption* that *all machine learning models make* is that all factors in the input data *relate, affect, and impact* the final outcome. This could cause problems with accuracy, and even lead to incorrect correlations. For example, if a model attempted to relate months, holidays and temperature to shopping activity, the computer may imply that cold temperatures lead to increased shopping activity due to the heightened shopping in December, rather than the approaching holiday of Christmas leading to higher shopping activity. Similarly, noise can affect the accuracy and simplicity of our model.

Figure 14: Removing noise can lead to increased accuracy



In the process of designing our model, we integrated *Yggdrasil Decision Forests* (YDF, n.d.), a Google-created machine learning library that allows for the creation of quick, out-of-the-box machine learning models. After naively feeding our data into a default YDF model, we tested the model and had it output the relative importance of certain variables:

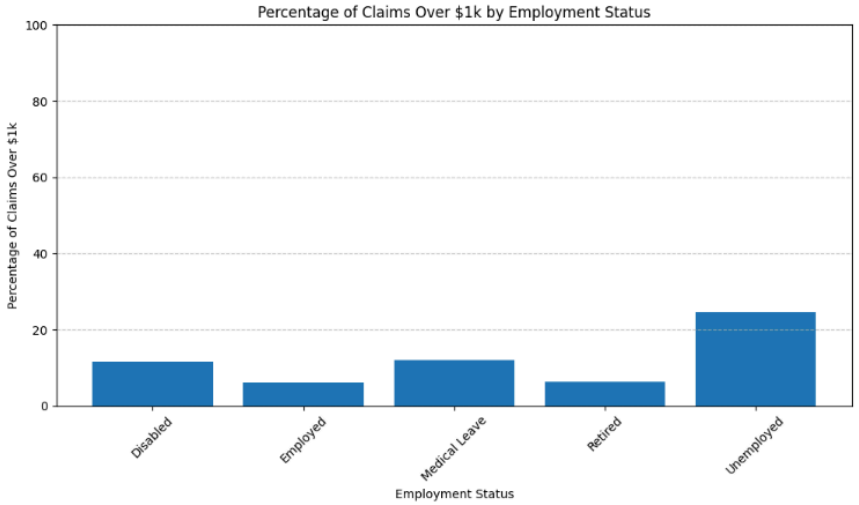
Figure 15: The output of the "analyze" function of YDF

```
 1.       "Customer Lifetime Value"  0.132748 ################
 2.            "Number of Policies"  0.105869 ############
 3.                "Marital Status"  0.027427 ###
 4.                        "Income"  0.014811 #
 5.             "Employment Status"  0.014262 #
 6.                      "Coverage"  0.012617 #
 7.                     "Education"  0.003291
 8.                        "Policy"  0.000549
 9.                  "Sales Channel"  0.000549
10.                   "Vehicle Size"  0.000549
11.                    "Policy Type"  0.000000
12.                           "Year"  0.000000
13.                          "Month" -0.000549
14.       "Months Since Last Claim" -0.001097
15.                          "State" -0.001646
16. "Months Since Policy Inception" -0.001646
17.              "Renew Offer Type" -0.001646
```

YDF found a strong correlation between the highest values on this list and whether a client would submit a claim in excess of $1000—in other words, the variables highest on the list were most impactful in helping the model make correct predictions. Conversely, the variables near the bottom of the list were found by YDF to have little to no bearing on predicting whether a client would submit a claim in excess of $1000.

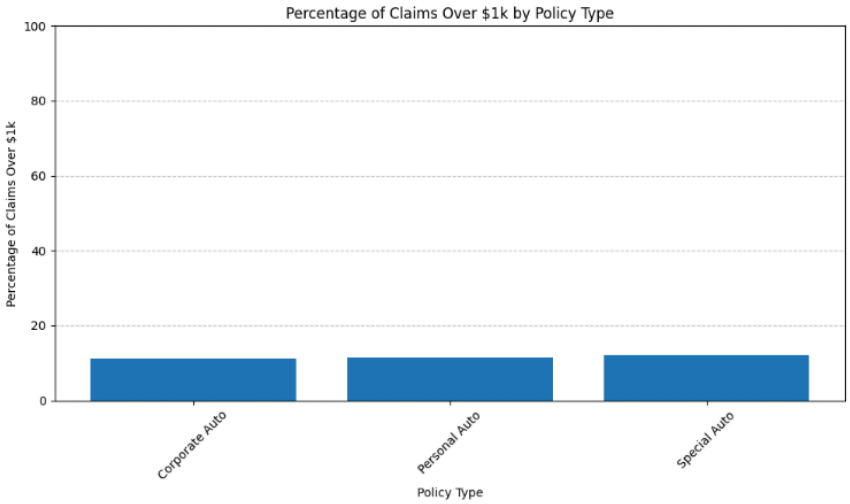
Note that the weighting of many of these variables makes sense on a qualitative level. We performed analysis on the data to find the fields that had strong correlation with claims over $1000. For example, employment status is flagged as an important field by YDF above, and we can see from the percentage of each employment class that filed a claim over $1000 that clients in certain employment classes, such as "unemployed" and "medical leave", have a higher chance of leading to a claim over $1000.

Figure 16: Percentage of Claims over $1000 by Employment Status



Conversely, fields like "policy type" that have been flagged by YDF as having little correlation can be likewise manually analyzed to find that clients with certain policy types are not more or less likely to file a claim over $1000:

Figure 17: Percentage of Claims Over $1000 by Policy Type



And so to a certain extent we can manually validate and qualitatively understand the results outputted by YDF. YDF is not entirely a black box.

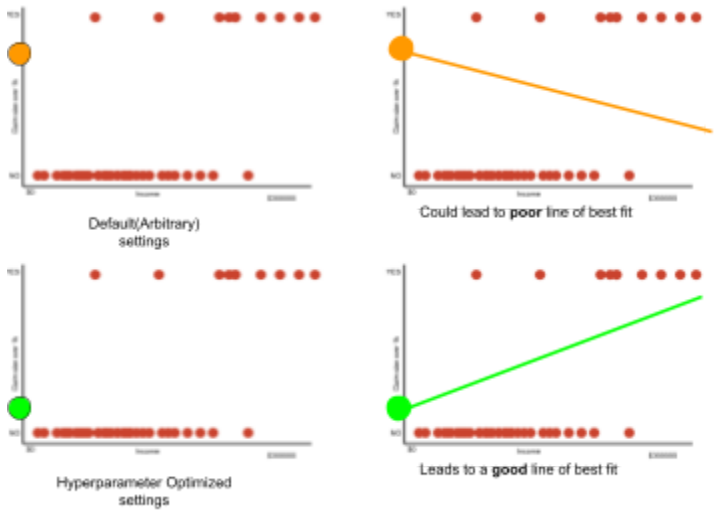
With this information in mind, we manually removed some of the variables at the bottom of YDF's output to test whether doing so would have a positive impact on predictive power, with the ultimate result that our final model removed noise variables such as Gender and Number of Open Complaints and achieved a higher accuracy as a result.

One may also question why we decided to use only four base models in our stacking model. Fundamentally, while this insurance dataset is large, it is not extremely complicated. A certain amount of input values correlate to an output value. Using more complex models and crowding our base model ensemble with even more models or perhaps integrating more complicated techniques such as deep learning would only serve to over complicate our model's decision making and clutter up its decision. Noise can be introduced and spurious relationships can be determined by such a model. Hence, we kept our design relatively simple and straightforward, rather than bolting on every cutting edge machine learning paradigm available.

Hyperparameter Optimization

Beyond data cleaning, our design also focused on modifying the hyperparameters of the models we utilized. Hyperparameters are the little knobs and dials built into the machine learning libraries we used, such as XGBoost. Changing these settings affects how the models train and thus their final accuracy and predictive power. For example, the XGBoost hyperparameter *eta* affects how much weight each successive decision tree that the model generates affects the overall prediction. Higher eta values lead to a model that trains faster and with higher accuracy, but can potentially lead to overfitting. Thus, hyperparameters allow a machine learning model to know "where to start" and "how to correct" over assuming arbitrary starting points.

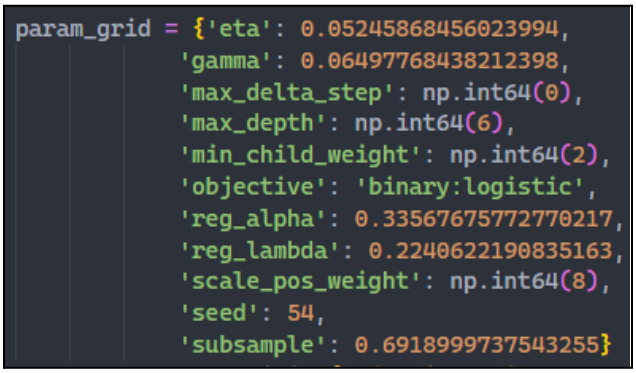
Figure 18: Hyperparameter optimization can be thought of as "setting a good starting point". As seen below, choosing a good starting point can have a big impact on making accurate predictions



Manually determining the optimal values for the several hyperparameters built into the models we utilized would have been an extremely time-consuming endeavor—hence, we automated the process via a library called *hyperopt* that tests multiple combinations of hyperparameters through techniques like random search (Bergstra, n.d.). For example, the final

hyperparameters found for XGBoost in the high sensitivity model found by hyperopt are the following:

Figure 19: the final hyperparameters for the high sensitivity model determined by hyperopt



With these two techniques, we were able to increase our stacking model's accuracy by 6.5%.

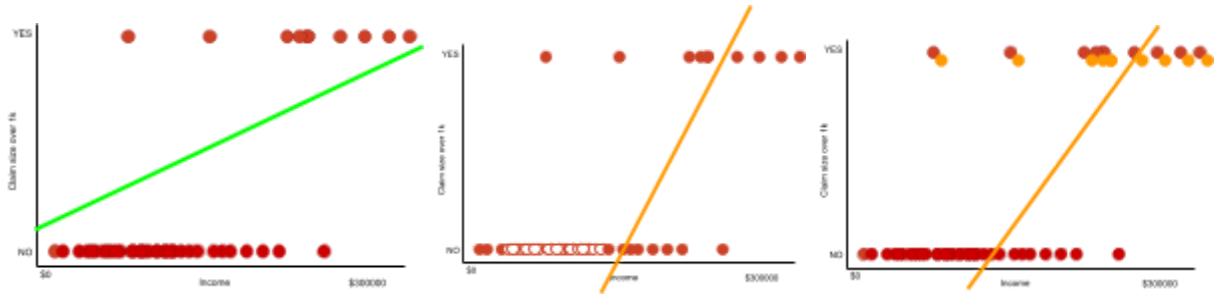
Figure 20: The increase in accuracy after optimization

Stacking model **without** optimization	Stacking model **with noise only** optimization	Stacking Model: **with noise and hyperparameter** optimization
88.1%	92.0%	**93.6%**

Oversampling and Undersampling

Another technique we attempted to further optimize the predictive power of our model was oversampling and undersampling. 88% of the training data provided consisted of clients who did not submit a claim over $1000, while 12% of the clients did. As a result, our models had a limited amount of data regarding what clients who file large claims look like and is termed *imbalance*, as there is far more data in the majority case (did not submit a claim over $1000) than in the minority case (submitted a claim over $1000). With less data comes less accuracy—and this is a common problem in machine learning. Two recommended techniques to combat this are oversampling and undersampling (Wu, 2022). The former involves synthetically creating more data in the minority class, while the latter involves synthetically removing certain data in the majority class. Both techniques operate with the common goal of creating a more balanced, 50/50 split between the minority and majority classes.

Figure 21: Undersampling (center diagram) and oversampling(right diagram) can lead to erroneous lines of best fit (left diagram)



However, after attempting these techniques, we found accuracy decreased. This is explained by the fact that our dataset is very small: undersampling decreases the amount of data even further, destroying the amount of data the model has to work with and thus its accuracy. Conversely, oversampling introduced more noise and magnified the errors already present in the dataset, as there was very little existing minority class data to build more out of. For this reason, we decided not to utilize oversampling and undersampling for our model.

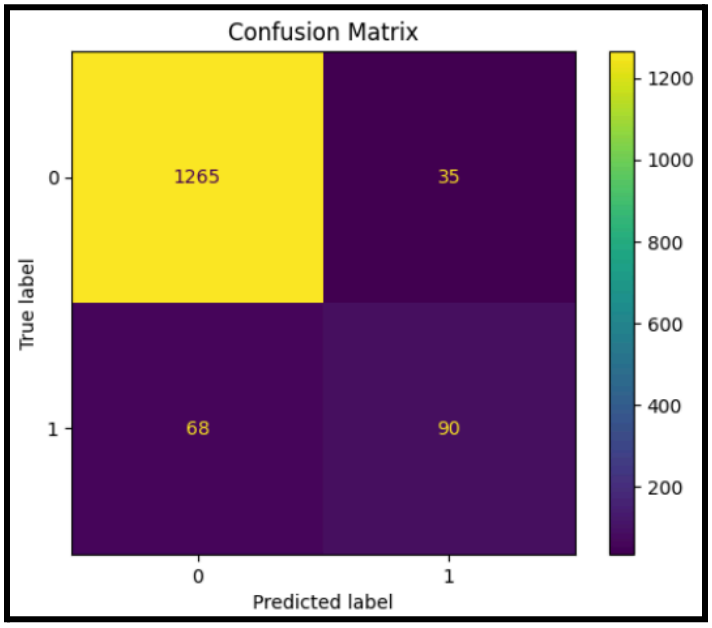
Figure 22: Statistical validation for undersampling and oversampling showed no gains in accuracy were made. For this reason, oversampling and undersampling were not used in the final model.

Stacking model **with undersampling**	Stacking model **with oversampling**	Stacking Model: **without undersampling or oversampling**
91.7%	92.9%	**93.6%**

Statistical Validation

In order to determine whether the above optimizations were valuable, a standardized and robust method of validating the accuracy of our model before and after modification had to be developed. To determine the accuracy of a single model, we examined the model's *confusion matrix*:

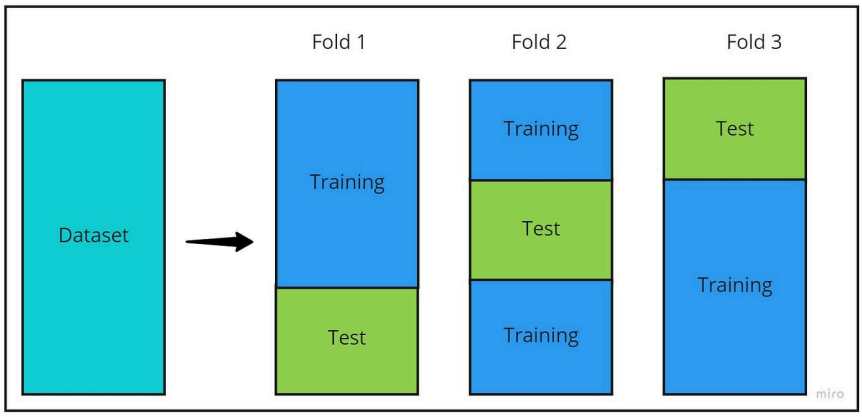
Figure 23: *A confusion matrix for a model.*



The top left corner shows the amount of test data where no claim over $1000 was filed, and which was categorized correctly. The top right displays the data of this form that was categorized incorrectly. Similarly, the bottom right corner shows the amount of test data where a claim over $1000 was filed, and which was categorized correctly, while the bottom left displays the data of that form which was categorized incorrectly. *Accuracy* is the percentage of correctly-categorized data. *Sensitivity* is the bottom right corner divided by the bottom row, i.e. the percentage of claims over $1000 that were categorized correctly. This is an important metric for insurance companies to pay attention to, as failing to categorize a claim over $1000 correctly can lead to more risks and potentially cause the company to have less assets on hand to cover the costs of claims. Our sensitivity model works to optimize this sensitivity value.

However, just because one model's confusion matrix has a higher accuracy or sensitivity statistic than another does not immediately mean it is a better model. When training a model, the dataset is split into a training subset and a testing subset. For our models, we utilized an 80/20 split to ensure a large amount of data was available for the models to train on while ensuring the test dataset provided accurate results. However, a model with certain hyperparameters and data pre-cleaning may perform better on certain splits of the dataset into training and testing subsets, but perform worse on others. Thus, we must examine how the model does on average over multiple splits of the data. This is called k-fold cross validation.

Figure 24: *a diagram showing an example of three-fold cross validation.*



Every time our model changed, we executed a 10-fold cross validation to test how it would do on our dataset on average, independent of the way our dataset was partitioned into training and testing subsets (See code in *Appendix E*). Our final accuracy and sensitivity values were an average of the accuracies and sensitivities in each of the ten cases. This methodology ensured that we could be confident in a gain or decrease in predictive power after implementing an optimization. For example, note the below list of accuracies has some variance. Depending on the partition of the dataset, we can expect the model to sometimes perform better and sometimes perform worse. However, on average, we can expect our high sensitivity model to have an accuracy around 89.6%.

Figure 25: *Accuracy of our sensitivity model across the 10 partitions of the dataset.*

```
[0.8921232876712328,
 0.8715753424657534,
 0.8801369863013698,
 0.8904109589041096,
 0.8955479452054794,
 0.8972602739726028,
 0.8818493150684932,
 0.9075342465753424,
 0.8886986301369864,
 0.9041095890410958]
```

Discussion

The machine learning models outlined above could potentially yield the following positive business impacts for an insurance company:

Enhanced Pricing Strategies

The primary application of our model is in underwriting, where it has been hyper-tuned to maintain an optimal balance between sensitivity and specificity, crucial given the dataset's size and imbalance. Achieving this balance required strategic adjustment, prioritizing sensitivity slightly over specificity to effectively capture high-risk cases. For instance, our stacking model achieved an accuracy above 93%, though it underperformed on true positive identification, with a sensitivity just over 50%. In contrast, an XGBoost model with YDF tuning achieved a sensitivity above 80%, though overall accuracy decreased by 4%. This balance positions the model as a powerful tool for categorizing clients into high-risk and low-risk groups, reducing chances of over- or under-prediction.

This categorization directly supports enhanced pricing strategies. While the model provides binary classifications (1 for high risk, 0 for low risk), these outputs are derived from nuanced analysis, allowing us to assign clients into specific risk levels. For example, a high-risk classification indicates clients who may not be eligible for premium pricing, whereas a low-risk classification identifies clients more suited for favorable rates. Aligning premiums with these classifications benefits both the insurer and the client, as high-risk clients are adequately priced while low-risk clients avoid unnecessary overcharging. This precision supports optimized profitability and fairness, positioning the model as an integral asset in underwriting and pricing decisions.

Cost and Wait Time Reduction through Claim Triage

The aforementioned challenges in the current insurance industry – specifically, the expensive underwriting process and lengthy processing times (Batty, n.d.) – could be significantly improved through our model's triaging process. This process could proceed as follows:

When receiving information from a new client, our predictive model generates a predictive score on a scale from 0 to 1, reflecting the application's quality. This output is then compared against a predetermined range. For instance, suppose the boundaries are set at 0.35 and 0.65. In this case, clients with scores below 0.35 are likely to be accepted, while those above 0.65 may be rejected, allowing for almost immediate feedback. Applications that fall between these boundaries would require traditional underwriting before a decision is reached.

Implementing this triaging process could lead to significant savings for insurers. According to the Society of Actuaries (SOA), typical company volumes suggest that annual savings from reduced underwriting requirements and faster processing could amount to millions, more than covering the model development cost (Chester, 2019). The table below illustrates potential annual savings for a representative life insurer, outlining standard underwriting requirements, associated costs, and usage frequencies in both traditional and model-enhanced processes, followed by a comparison of underwriting costs for each approach.

Figure 26: *Potential financial impact after integrating the machine learning model.* (Batty, n.d.)

	Requirement Cost	Requirement Utilization	
		Traditional Underwriting	Predictive Model
Paramedical Exam	$55	50%	0%
Oral Fluids Analysis	$25	20%	0%
Blood and Urine Analysis	$55	70%	0%
MVR Report	$6	70%	75%
Attending Physician Statement	$100	20%	0%
Medical Exam	$120	20%	0%
EKG	$75	10%	0%
Stress Test	$450	1%	0%
Third-Party Data	$0.50	0%	100%
Total Cost Per Applicant		**$130**	**$5**
Savings Per Applicant		**$125**	
Annual Applications Received		**50,000**	
Annual Savings (over 30% to 50% of applications)		**$2 to $3 million**	

Additionally, a preliminary selection process would streamline the process time for many clients. The longer an application process lasts—and the more tests an applicant must undergo—the greater the likelihood that the applicant will become frustrated, potentially abandoning the purchase or accepting an offer from a competitor (Kagan, n.d.). Our predictive model is highly efficient, producing meaningful outcomes almost instantly for a large group of clients, thus enhancing their experience. This agile handling of applications also frees up increasingly scarce underwriting staff to manage larger volumes, as the model takes on more routine tasks.

Identification of New Market Opportunities

Third, our YPT model could help identify trends that reveal population segments likely to become high-quality clients. As a result, the company could launch targeted promotional campaigns to attract more desirable clients. This advantage stems from our model's strength in analyzing hidden relationships within the data provided. In its implementation, illustrated in figure 15, the model presents a thorough analysis of the correlation between various data categories and the probability of making a claim exceeding $1,000 in auto insurance. For instance, factors such as gender contribute minimally to accurate predictions, and in fact, they introduce noise that disrupts data clarity and lowers overall accuracy. By leveraging machine learning to assess the importance of

each data category, our model generates valuable insights, helping companies better understand the ideal client profiles to target. For instance, our YPT model identifies Marital status, Income, and Employment Status as the top factors correlated with high claim probability, highlighting these attributes as key for targeting. In contrast, attributes such as education level, car type, and location have lower predictive value, suggesting that clients within these categories tend to have fewer or smaller claims. This insight presents a profitable demographic to pursue through a targeted advertising campaign.

Additionally, the ability to identify low-risk clients and offer them lower premiums strengthens the foundation for attracting and retaining valuable clients. Companies can further leverage these insights by cross-selling or upselling additional products to targeted low-risk clients, thereby increasing their lifetime value. In *Appendix F*, the model demonstrates its accuracy, predicting 251 claims over $1,000, compared to the actual count of 166. This conservative prediction enables the company to make more informed decisions and prepare for high-value claims by setting up contingency measures, such as an emergency fund, to ensure sufficient cash flow for large claims. These proactive steps not only enhance financial stability but also improve the efficiency of claim processing.

Strengthened Fraud Detection

Fraud represents a significant risk for insurance companies, with auto fraud alone costing U.S. insurers approximately $29 billion annually (*Background on: Insurance Fraud | III*, 2022). Fraudulent activities can vary widely, from exaggerated repair costs to serious offenses like staged accidents and false theft claims. To combat this, our stacking model offers a valuable tool for fraud detection and efficient claims processing. By identifying claims that deviate from typical patterns, our model allows insurers to proactively investigate potentially fraudulent high-value claims. Our stacking model is specifically hyper-tuned for accuracy. While accuracy-focused models may have slightly lower sensitivity, this design choice effectively reduces false positives (instances where claims are predicted to exceed $1,000 when they do not). According to the confusion matrix in the *Appendix G*, our model has achieved a low false-positive rate of 0.016% while maintaining a sensitivity accuracy of 56%. Insurance companies can thus rely on this model to flag high-value claims that appear inconsistent with predicted patterns, ensuring cases warranting further investigation are identified early.

Moreover, for claims exceeding $1,000 that align with our model's predictions, insurers can implement a streamlined process tailored to expedite these cases, freeing resources and reducing administrative efforts. This two-pronged approach—thorough review for flagged cases and streamlined processing for consistent ones—can significantly improve claim integrity and processing efficiency, ultimately enhancing client satisfaction by enabling accurate, timely claim management.

Together, these enhancements position our model as a transformative tool for strategic pricing, operational efficiency, market growth, and fraud prevention, fostering a more agile and client-centric insurance business.

Limitations

The limitations of this paper largely stem from the characteristics of the testing dataset provided. A prominent challenge is the inherent imbalance in auto claims, where the ratio of claims over $1,000 to those under $1,000 in the dataset provided is nearly 1:8. This imbalance skews machine learning performance (Awe, n.d.), as the model becomes more inclined to predict the majority class (claims under $1,000) while underperforming in predicting the minority class (claims over $1,000). Consequently, this class imbalance reduces the model's sensitivity and limits its effectiveness in accurately identifying cases that are likely to result in claims exceeding $1,000.

Additionally, while the provided data is cleaned, there are potentially useful features that are not included, such as the driver's experience level, distance between home and workplace, total insurance amount claimed, and fraud indicator indexes. These additional features could reveal valuable insights into factors contributing to high claims, potentially improving the model's predictive accuracy. Furthermore, despite the cleaning process, undetected instances of fraudulent claims may still exist in the dataset, potentially inflating the false positive rate in our predictions. With increased efforts in fraud detection, we are confident that the dataset could better reflect true outcomes, enhancing overall accuracy.

Lastly, the dataset is simply limited in scope. A larger dataset with a wider variety of cases, including both claims above and below $1,000, would allow the model to train on a broader range of scenarios. This diversity would improve the model's ability to identify meaningful correlations and produce more accurate predictions.

Potential Extension

When additional resources are available, several potential extensions could further improve our predictive model's accuracy and reliability:

Firstly, the simplest way to improve model accuracy is by expanding the dataset with more factors that correlate positively with the prediction of claims over $1,000. For instance, insurance companies could gather additional information for each automobile claim by requesting clients to provide more detailed data in their applications. Data points like driver experience, collision specifics, number of prior collisions, vehicle price, and age may reveal new insights and correlations, boosting the predictive power of our model.

Secondly, implementing a deep learning model utilizing neural networks could greatly enhance accuracy once sufficient data is collected. Deep learning models are well-suited for tasks involving large datasets and complex patterns (Holdsworth, 2024) because of their multi-layered approach to automatically learn intricate features. However, due to our current data limitations, this is a longer-term extension. With adequate data, deep learning could significantly improve the model's accuracy and sensitivity by identifying patterns traditional machine learning models might miss.

Finally, developing customized machine learning models rather than relying solely on out-of-the-box libraries could provide a more tailored approach. Custom models can be optimized to meet our specific requirements, allowing for more precise adjustments to improve performance in line with the unique characteristics of the auto claims dataset.

Conclusion

In this report, two machine learning models were leveraged to help actuarial firms process auto claims. Through the assessment of industry challenges and the outlining of focused metrics, we offer a structured approach to claims assessment. By detailing the methodology, implementation, and performance metrics, we have established a model that not only addresses current needs but also forecasts significant business impact. Additionally, the report outlines actionable steps for firms to achieve meaningful business outcomes, acknowledges existing model limitations, and proposes future enhancements to improve predictive accuracy and expand functionality, positioning this model as a valuable tool for advancing claims analysis in the industry.

As machine learning continues to evolve, its influence on traditional actuarial roles will deepen, sparking debates over the value of clinical versus actuarial judgment in key processes like underwriting. Actuaries, with their "trained minds," should pursue the practical over the intuitive and prioritize the actuarial over the clinical, because, as W. Edwards Deming put it:

Without data, you're just another person with an opinion.

Appendices

Appendix A:

Code on cleaning and splitting the data:

```
# renaming columns
df.columns = df.columns.str.replace(' ', '_')

# drops
df.drop(['CustomerID', 'Coverage', 'Education', 'Employment_Status',
'Gender', 'Number_of_Open_Complaints', 'Marital_Status', 'Policy_Type',
'Policy', 'Sales_Channel', 'Vehicle_Size'], axis=1, inplace=True)

# continuous date
df['Year'] = pd.to_datetime(df['Effective_To_Date']).dt.year
df['Month'] = pd.to_datetime(df['Effective_To_Date']).dt.month
df['Effective_To_Date'] = df['Year'] * 0.365 + df['Month'] * 0.030
df.drop(['Year', 'Month'], axis=1, inplace=True)
```

Appendix B:

Code used in “one-hot-encoding”, to ensure that discrete data is properly employed in machine-learning algorithms:

```
# one-hot encoding
X_encoded = pd.get_dummies(X, columns=[
    'State',
    'Response',
    'Coverage_Index',
    'Education_Index',
    'Employment_Status_Index',
    'Marital_Status_Index',
    'Policy_Index',
    'Policy_Type_Index',
    'Sales_Channel_Index',
    'Vehicle_Size_Index',
    'Renew_Offer_Type'
  ])
```

Appendix C:

Given a dataset $\{\boldsymbol{D}, \boldsymbol{y}\}$ and p CARTs $f(\boldsymbol{x})$ as weak learners, the ensemble $F_0(\boldsymbol{x})$ first includes a weak learner $f0\ (\boldsymbol{x})$ that learns from the original dataset. Then, the ensemble sequentially adds weak learners that learn from the residual of the previous ensemble. If $t > 0, t \in N$ is the t-th boosting round, then the ensemble $Ft(\boldsymbol{x})$ at the t-th boosting round is:

$$F_t(\boldsymbol{x}) = \sum_{i=0}^{t} f_i(\boldsymbol{x})$$

Where $ft\ (\boldsymbol{x})$ learns from the residuals of $Ft{-}1(\boldsymbol{x})$, and is the learner that greedily minimizes an objective function L:

$$L^t = \sum_{i=1}^{n} l(y_i, F_{t-1}(\boldsymbol{x_i}) + f_t(\boldsymbol{x_i})) + \Omega(f_t)$$
$$\Omega(f_t) = \gamma T + \frac{\lambda \|w\|^2}{2}$$

Where l is a differentiable complex loss function between the i-th outcome yi and the $(t-1)$-th ensemble's predicted i -the outcome $F_{x\text{-}1}(\boldsymbol{xi})$, and $\Omega(ft)$ is a function that penalizes tree complexity, with T, w as the amount of leaves and sum of all leaf weights respectively, and γ, λ respectively are the regularization and minimum loss hyperparameters of XGBoost.

As with gradient tree boosting machine learning algorithms, XGBoost can (locally, in a given ensemble) calculate the importance of variables in a dataset [15, 24]. Given a variable V in a CART, the improvement I(V) of a variable that splits a parent node P into child nodes L, R, of which q is the fraction of paths that pass-through L is defined by:

$$I(V) = E(P) - (qE(L) - (1-q)E(R))$$

Where $E(K)$ is the weighted squared errors of node K. Importance of a variable in an ensemble is defined as the average of improvement of said variable of all trees in the ensemble (Fauzan, n.d.)

Appendix D:

The code to stack models is as follows:

```
# Define base models for stacking
base_models = [
    ('dt', DecisionTreeClassifier(random_state=42, ccp_alpha=0.001,
max_depth=7, min_samples_leaf=5)),
    ('svc', SVC(random_state=42, C=150, gamma=0.0005, kernel='rbf')),  #
Missing comma added here
    ('xgb_base', XGBClassifier(gamma=0, learning_rate=0.12, max_depth=5,
reg_lambda=0, scale_pos_weight=1))
    ('ydf', YDFClassifier(gamma=0, random_state=0.12, max_depth=8,
scale_pos_weight=1))
]
meta_model = RandomForestClassifier(n_estimators=100, max_depth=5,
min_samples_split=2)

stacking_model = StackingClassifier(
    estimators=base_models,
    final_estimator=meta_model,
    cv=5
)


stacking_model.fit(X_train, y_train)

y_pred = stacking_model.predict(X_test)
```

Appendix E:

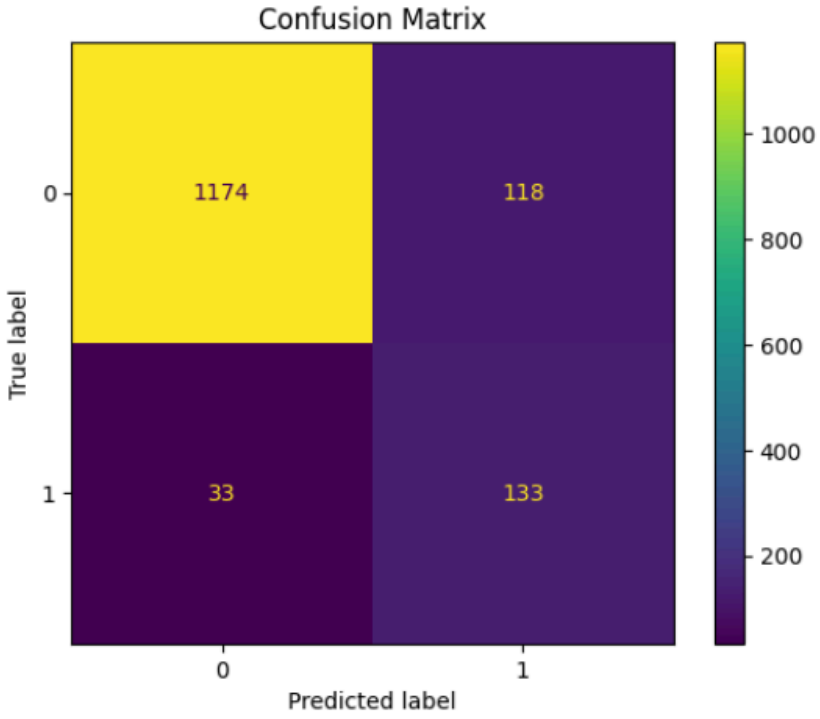
Statistical Validation: 10-Fold Cross Validation Code

```
Sss = StratifiedShuffleSplit(n_splits=10)
sss.get_n_splits(X_train, y_train)

for train_index, test_index in sss.split(X_train, y_train):
        X_train_cv, X_test_cv = X_train.iloc[train_index],
X_train.iloc[test_index]
        y_train_cv, y_test_cv = y_train.iloc[train_index],
y_train.iloc[test_index]
        clf_xgb.fit(X_train_cv, y_train_cv)
        matr = confusion_matrix(y_test_cv, clf_xgb.predict(X_test_cv))
        scores.append((matr[0][0] + matr[1][1]) / (matr[0][0] + matr[0][1]
+ matr[1][0] + matr[1][1]))
```

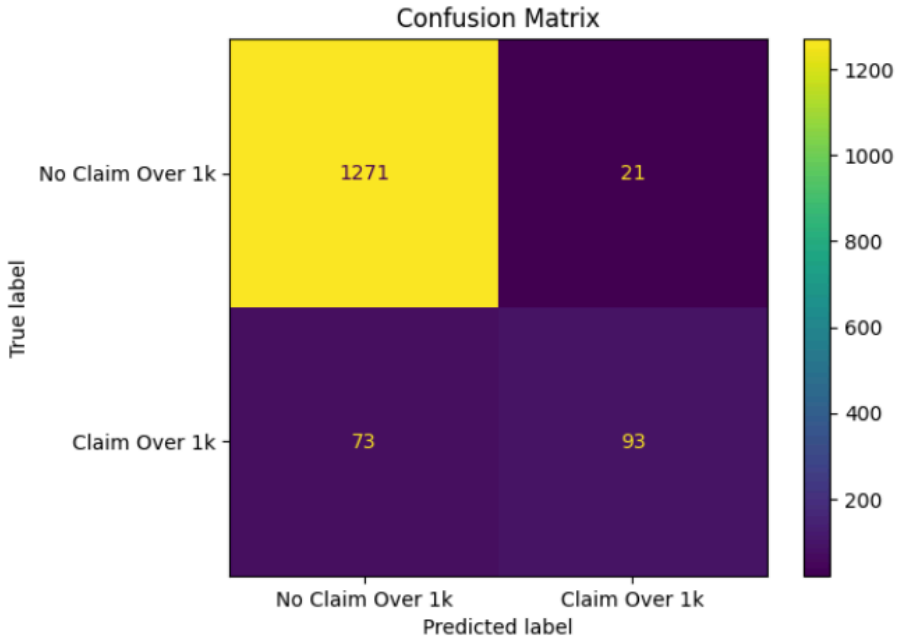
Appendix F:

Confusion Matrix for the XGBoost Model



Appendix G:

Confusion Matrix for the Stacking Model



Appendix H:

Code for Splitting Dataset into Training and Testing

```
X_train, X_test, y_train, y_test = train_test_split(Xs[0], ytr,
test_size=0.2, stratify=ytr, shuffle=True)
```

References

1. Alamir, E. (2021). Motor Insurance Claim Status Prediction using Machine Learning Techniques. https://www.researchgate.net/profile/Endalew-Alamir/publication/350550914_Motor_Insurance_Claim_Status_Prediction_using_Machine_Learning_Techniques/links/61adfdf6ca2d401f27cd9e91/Motor-Insurance-Claim-Status-Prediction-using-Machine-Learning-Techniques.pdf
2. Awe, O. O. (n.d.). *Handling Data Imbalance in Machine Learning*. ISI. Retrieved November 1, 2024, from https://www.isi-web.org/sites/default/files/2024-02/Handling-Data-Imbalance-in-Machine-Learning.pdf
3. *Background on: Insurance fraud | III*. (2022, August 1). Insurance Information Institute. Retrieved November 1, 2024, from https://www.iii.org/article/background-on-insurance-fraud
4. Batty, M. (n.d.). *Predictive Modeling for Life Insurance Prepared by Deloitte Consulting LLP*. SOA. Retrieved November 1, 2024, from https://www.soa.org/globalassets/assets/files/research/projects/research-pred-mod-life-batty.pdf
5. Bergstra, J. (n.d.). Algorithms for Hyper-Parameter Optimization. https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
6. Brown, S. (2021, April 21). *Machine learning, explained*. MIT Sloan. Retrieved November 1, 2024, from https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained
7. Chester, A. (2019, February 12). *From art to science: The future of underwriting in commercial P&C insurance*. McKinsey & Company. Retrieved November 1, 2024, from https://www.mckinsey.com/industries/financial-services/our-insights/from-art-to-science-the-future-of-underwriting-in-commercial-p-and-c-insurance
8. Fauzan, M. A. (n.d.). The Accuracy of XGBoost for Insurance Claim Prediction. (ISSN 2074-8523), 6 - 7. https://www.i-csrs.org/Volumes/ijasca/11_IJASCA_The-accuracy-of-XGBoost_159-171.pdf

9. Gillis, A. S. (n.d.). *What is data splitting and why is it important?* TechTarget. Retrieved November 1, 2024, from https://www.techtarget.com/searchenterpriseai/definition/data-splitting
10. Hearst, M. A. (n.d.). Support vector machines. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=708428&tag=1
11. Holdsworth, J. (2024, June 17). *What Is Deep Learning?* IBM. Retrieved November 1, 2024, from https://www.ibm.com/topics/deep-learning
12. Jain, S. (2024, March 21). *One Hot Encoding in Machine Learning*. GeeksforGeeks. Retrieved November 1, 2024, from https://www.geeksforgeeks.org/ml-one-hot-encoding/
13. Kagan, J. (n.d.). *Waiting Period: Definition, Types, and Examples*. Investopedia. Retrieved November 1, 2024, from https://www.investopedia.com/terms/w/waiting-period.asp
14. Promutuel Insurance. (n.d.). *Insurance risk assessment: How does it work? | Promutuel Insurance*. Promutuel Assurance. Retrieved November 1, 2024, from https://www.promutuelassurance.ca/en/blog/insurance-risk-assessment-how-does-it-work
15. Samson, D. (n.d.). Linear models as aids in insurance decision making: The estimation of automobile insurance claims. https://www.sciencedirect.com/science/article/pii/0148296387900270
16. scikit. (n.d.). *1.10. Decision Trees — scikit-learn 1.5.2 documentation*. Scikit-learn. Retrieved November 1, 2024, from https://scikit-learn.org/1.5/modules/tree.html
17. *Sensitivity ,Specificity, and Accuracy: Understanding Model Performance*. (2024, February 22). Analytics Vidhya. Retrieved November 1, 2024, from https://www.analyticsvidhya.com/blog/2021/06/classification-problem-relation-between-sensitivity-specificity-and-accuracy/
18. Society of Actuaries (Director). (2020). *The Evolution of Predictive Models in Life Insurance Underwriting* [Film]. https://www.youtube.com/watch?v=B3fUYG3-JbQ&t=2082s
19. Tiemann, K. (2019, January 8). *How Much do Insurance Companies Spend on Marketing?* Leadsurance. Retrieved November 1, 2024, from https://leadsurance.com/how-much-do-insurance-companies-spend-on-marketing/
20. Walach, E., McDavid, E., Vajani, M., & Baum, S. (2021, September 21). *Sensitivity vs. specificity: The eternal AI debate*. MedCity News. Retrieved November 1, 2024, from https://medcitynews.com/2021/09/sensitivity-vs-specificity-the-eternal-ai-debate/

21. Wu, Y. (2022, August 24). *7 Techniques to Handle Imbalanced Data*. KDnuggets. Retrieved November 1, 2024, from https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html
22. XGBoost. (n.d.). XGBoost Documentation — xgboost 2.1.1 documentation. Retrieved November 1, 2024, from https://xgboost.readthedocs.io/en/stable/
23. YDF. (n.d.). YDF documentation - YDF documentation. Retrieved November 1, 2024, from https://ydf.readthedocs.io/en/stable/
24. Yong, E. (2013, January 31). *The Real Wisdom of the Crowds*. National Geographic. Retrieved November 1, 2024, from https://www.nationalgeographic.com/science/article/the-real-wisdom-of-the-crowds
25. Zangre, A. (2024, September 13). *Discrete vs. Continuous Data: What's the Difference?* G2. Retrieved November 1, 2024, from https://www.g2.com/articles/discrete-vs-continuous-data