ASNA Hackathon
Second Round Submission

Agile Insurance

Joey Shum
Noah Nguyen
Quang Bui
Yuma Teshirogi

November 1st, 2024

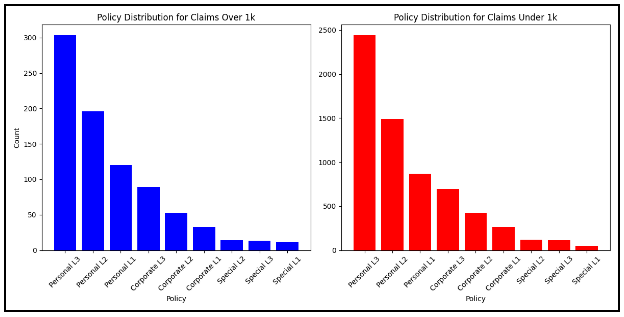
Contents

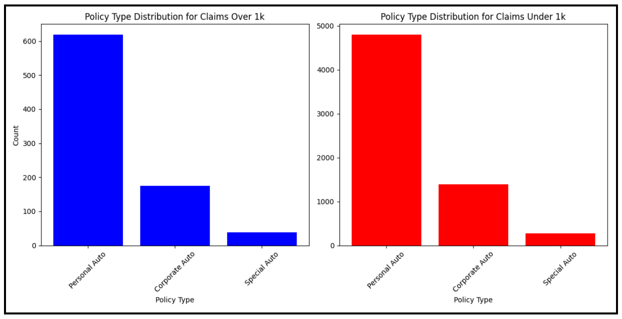
1. Executive Summary

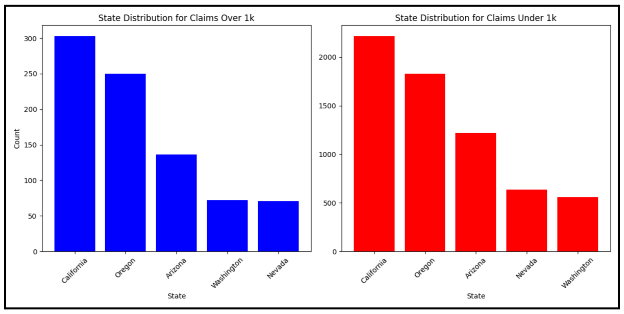
This report aims for the approval of the Machine Learning (ML) model developed by Agile Insurance, including a high-level explanation of key components of the model, as well as the estimated impact on the business. The construction and analysis of this model is based on the 9,134 records of auto insurance claims provided by the ASNA Hackathon team, including attributes relating to auto insurance policyholders. We will touch upon key attributes further into the report. Our primary objective is to predict the claims over $1000, utilizing an accurate model that also has high explainability.

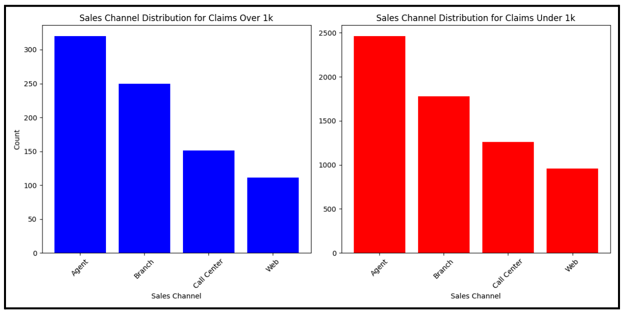
2. Methodology

2.1. Exploratory Data Analysis (EDA)

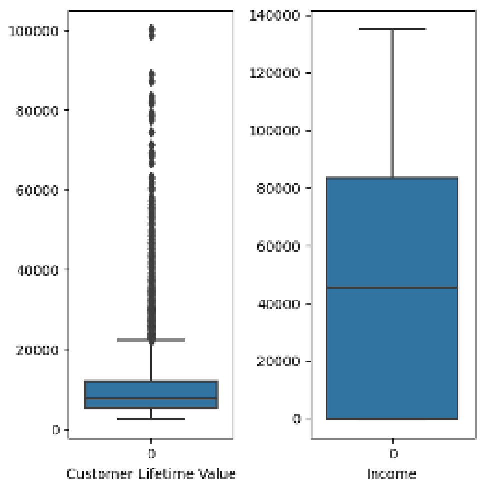








Looking at the distribution patterns across different features (Policy Type, Policy, State, and Sales Channel) for both classes, there is a notable similarity in their distributional shapes. Both classes follow similar relative proportions and patterns across all categories within each feature, just with different scales. This parallel distribution pattern suggests that these features may have limited discriminative power for classification purposes, as they don't show distinct patterns that could effectively separate class 0 from class 1.



The 2 continuous numerical features show distinctly different distributions. Customer Lifetime Value has many significant outliers stretching far above the main distribution, while Income shows a more balanced distribution without outliers. This contrast in distributions suggests that Customer Lifetime Value may need special preprocessing such as outlier treatment before it is incorporated into the model.

2.2. Data Preprocessing

Handling Categorical Variables
The dataset includes several categorical columns with overlapping information, where each category has both a name column and an assigned index. An instance of this is where the "Education" column has the description of level of education, and the "Education Index" column has the index corresponding to it. For ordinal variables we use the index columns, while for nominal variables we apply one-hot encoding and remove their index columns. This approach is used for nominal columns such as "Employment Status", "Marital Status", "Response" and "Gender". Additionally, we converted the "Effective To Date" column from a string to a datetime data type to facilitate future calculations and feature engineering.

Next, we scaled continuous features to improve model performance and stability. "Income" and "Average CLV" (to be added later) will use a Standard Scaler, standardizing values by removing the mean and scaling to unit variance, as these features do not have many outliers from the EDA section. "Customer Lifetime Value" and "Expected Claim Size" (another feature to be added later) will use a Robust Scaler, which is better suited for handling outliers by scaling based on the median and interquartile range. This tailored scaling approach minimizes biases from large numerical disparities, enhancing the model’s learning process.

2.3. Feature Engineering

2.3.1 Why do we want to overfit the model initially?

Overfitting the training data initially can help us understand the capacity of our model, revealing its upper limits and potential performance on the training set without regularization or other constraints. More specifically, it can help us with the following things:

1. Baseline: Establish a baseline performance for the model.
2. Model Complexity Check: Allowing the model to overfit helps confirm that it has sufficient capacity to capture patterns in the data. If it’s unable to overfit the training data, the model may be too simple, or the limitation could stem from preprocessing or data quality issues.

We will attempt to overfit the model by adding more features through feature engineering, guided by our intuition and business insights.

2.3.2 Engineering

In this section, we will describe the methodology of our feature engineering, as well as the rationale we had when designing these variables. Note that for each feature, we have proven that using these improves the model's predictability through adequate tests.

1. **Average CLV**

Calculation:
This column calculates the average Customer Lifetime Value (CLV) by dividing CLV by the number of months since policy inception.

Rationale:
CLV is the total value the insurer can get from the customer, which takes premiums into account. A riskier customer will have to pay higher premiums because they have a higher probability to make a claim, causing a loss for the insurer.
There is no premium information in the data set provided, but the average CLV can be used as an estimate. CLV takes into account the total premium received from the customer, so it is possible that a low risk customer who has been with the same insurer for a longer time will have a higher CLV than a high risk customer who is new with the insurer. Accounting for this, we have adjusted CLV based on the number of months since policy inception.

2. **Average Complaints**

Calculation:
This column divides the number of complaints by the number of policies.

Rationale:
Some customers could have multiple policies, so the total number of complaints is not insightful for us if we do not consider the number of policies an individual has. For example, if a customer with 8 policies only has 2 complaints, that is a relatively low complaint rate. However, if a customer has 2 policies and 2 complaints, they have 1 complaint per policy on average. This complaint rate is relatively high. This logic motivated us to account for the number of policies in order to obtain a more accurate complaint rate.

3. **Policy Age (in months)**

Calculation:
This column calculates the number of months between Current Date and Effective To Date columns. Current date is assumed to be 10/1/2024, assuming that the data set is retrieved as of 10/1/2024.

Rationale:
If the original Effective To Date column was left untouched, it could be formatted as a categorical column, but it wouldn't be very useful since there are too many categories. Instead, we convert it to a numerical column for easier interpretation. Month is chosen as the duration to match with the other time-related columns, which also have months as units.

4. **Expected Claim Size with Tweedie regression**

Calculation:
The Expected Claim Size is calculated as an estimation of the average claim amount a policyholder might make, based on a combination of historical data (Months Since Last Claim, Months Since Policy Inception), risk factors (number of policies, CLV), and advanced statistical techniques (Tweedie regression). The formula used to calculate the expected claim of a person is:

$$\text{Expected Claim Size} = \text{CLV} \times \left(\frac{\text{Months Since Last Claim}}{\text{Months Since Inception}}\right) \times \log(1 + \text{ Policies})$$

Rationale:
In this formula for the Expected Claim Size, three things were considered:

1. The long-term value of a customer indicates the value generated for the company. Higher CLV indicates increased policy engagement, longer tenure, all of which could imply a greater likelihood of claims or higher claim amounts. A customer with a higher CLV generally has a proportionally higher Expected Claim Size.

2. The recency of claim ratio (Months since last claim / Months since policy inception) indicates the recency of claims relative to how long the policy has been active. This effectively represents the claim frequency factor. The ratio serves to adjust the CLV by factoring in how recently a customer made a claim compared to their time with the policy. Customers who have recently made claims may indicate higher engagement with the policy. Conversely, if it has been a long time since the last claim relative to the policy duration, the Expected Claim Size might be adjusted downward, suggesting a potentially lower risk.

3. The Policy volume adjustment (the log function) represents the adjustment of the number of policies a customer holds, with a log transformation to reflect diminishing returns. Customers holding multiple policies could be seen as having more claims, but adding more policies doesn't increase risk linearly. Therefore, the logarithmic function applies a dampening effect, thus increasing Expected Claim Size in a way that balances growth without overemphasizing additional policies.
   This choice of function ensures that the contribution of multiple policies is incremental, but not exponential. Each additional policy held by the customer slightly increases the Expected Claim Size but with reduced influence, aligning with real-world observations that higher number of policies can increase risk but with diminishing impact.

2.4. Modelling

2.4.1 Evaluation Metrics:

Metrics and learning curves are obtained using 10-fold stratified cross-validation. For the learning curves, the training size progressively increases from 10% to 100% of the data, illustrating how the model's performance evolves with more training data. This technique is valuable for detecting underfitting or overfitting and for assessing how well the model generalizes as it learns from additional data. The confusion matrix, however, is evaluated based on the result from the first fold.

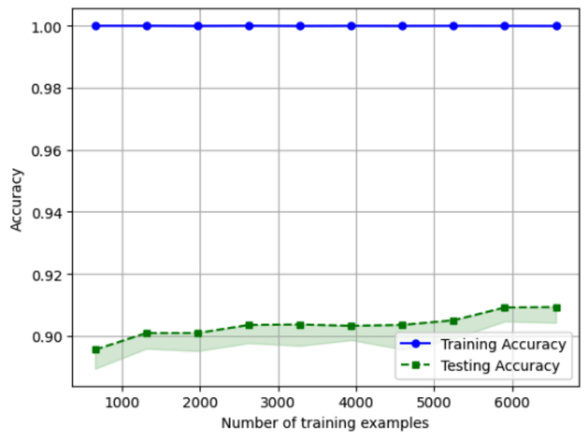
Interpretation of metrics:

- Recall: Measures the proportion of actual class 1 (over 1k) correctly identified by the model.
- F1 Score: The harmonic mean of precision (proportion of correctly predicted class 1 every time we predict class 1) and recall, balancing both metrics to handle class imbalances.
- ROC-AUC: Represents the model's ability to distinguish between classes, with higher values indicating better classification.
- Accuracy: The ratio of correctly predicted instances to the total instances, showing overall model correctness.

2.4.2 Baseline models:

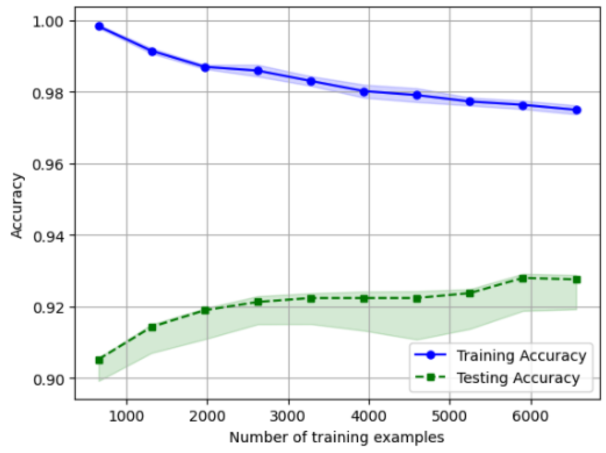
Algorithm \ Metric	Recall	F1 Score	ROC-AUC	PR-AUC
Random Forest	0.2668	0.4053	0.8961	0.9082
CatBoost	0.5107	0.6154	0.9190	0.9276
XG Boost	0.4974	0.5848	0.9002	0.9202
Light GBM	0.5204	0.6153	0.9103	0.9263

Random Forest:



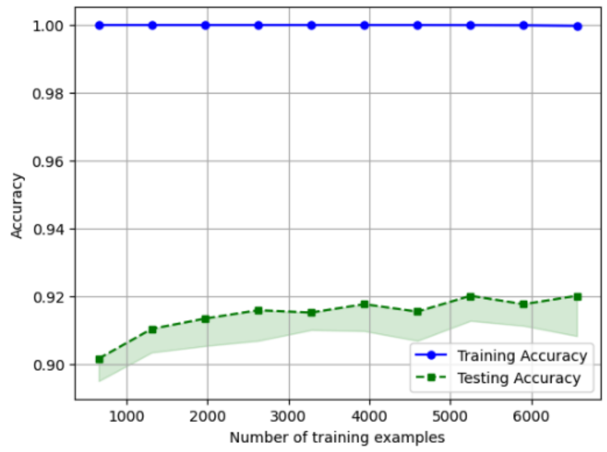
Confusion Matrix	Predict 0	Predict 1
Actual 0	640	6
Actual 1	56	27

CatBoost:



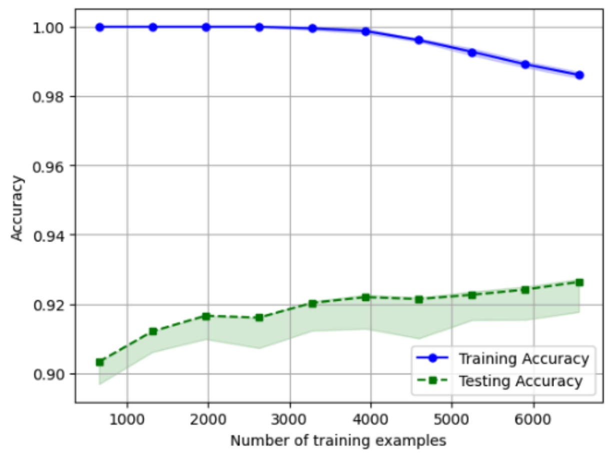
Confusion Matrix	Predict 0	Predict 1
Actual 0	630	10
Actual 1	34	49

XG Boost:



Confusion Matrix	Predict 0	Predict 1
Actual 0	633	13
Actual 1	32	51

Light GBM:



Confusion Matrix	Predict 0	Predict 1
Actual 0	631	15
Actual 1	30	53

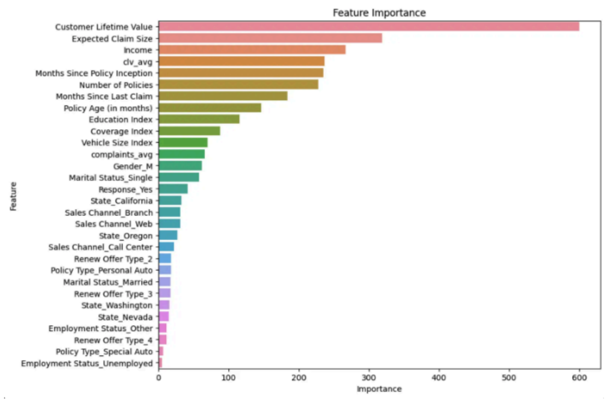
Insights:
The learning curve shows that the models are overfitting, so we will apply feature reduction techniques, regularization, and hyperparameter tuning to address this. Additionally, Random Forest performs significantly worse than the other three algorithms due to its difficulty in capturing class 1 accurately; therefore, we will not proceed with it.

2.4.3 Feature selection:

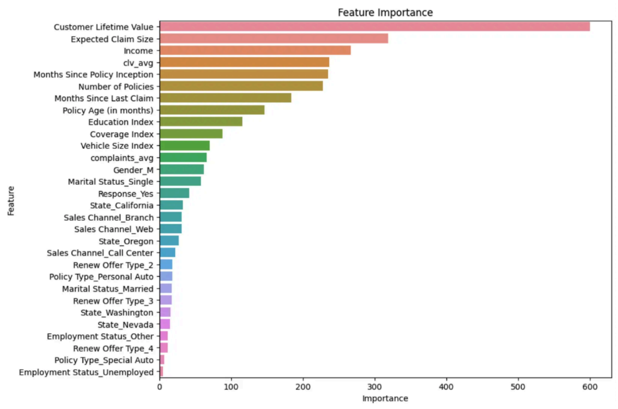
A SHAP (SHapley Additive exPlanations) chart visually explains how each feature contributes to a model's predictions by showing the impact of each feature on the output.

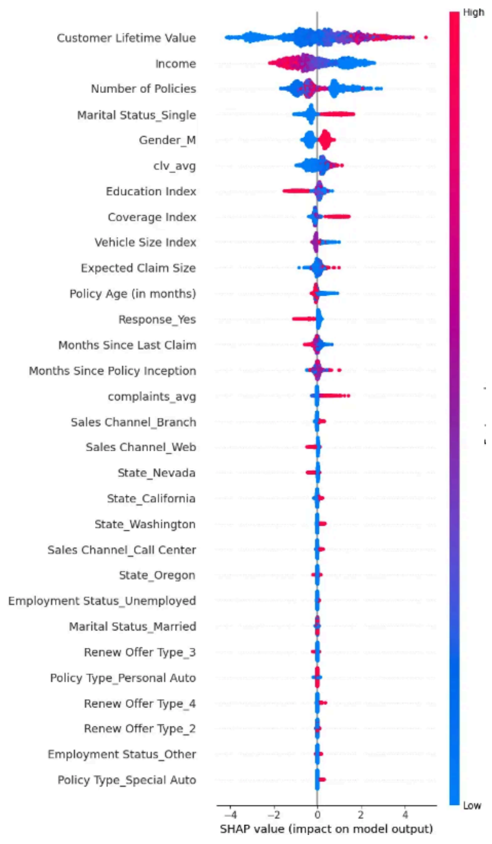
A feature importance chart for a boosting tree model shows which features most significantly influence predictions by ranking them based on their contribution to the model's decision process.

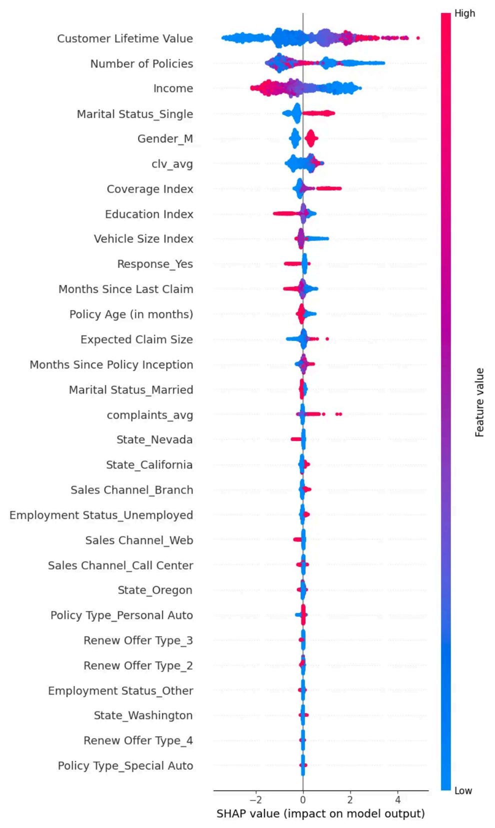
LGBM:



CatBoost:







The plot for XGBoost is also very similar, so we would only show the ones from LGBM and CatBoost here.

How can a SHAP chart be interpreted?
A SHAP chart makes a model more transparent by explaining how the variables affect the prediction. The use of different colours illustrate the feature values, and magnitude illustrates the impact of a feature on the prediction. For example, in our graph, CLV has the highest magnitude, so it is the most important feature. From the colours, we can see low values of CLV have a negative impact on the prediction, while high values of CLV have a positive impact on the prediction. This shows that policyholders with a higher CLV generally have a higher chance of making a claim over 1k.

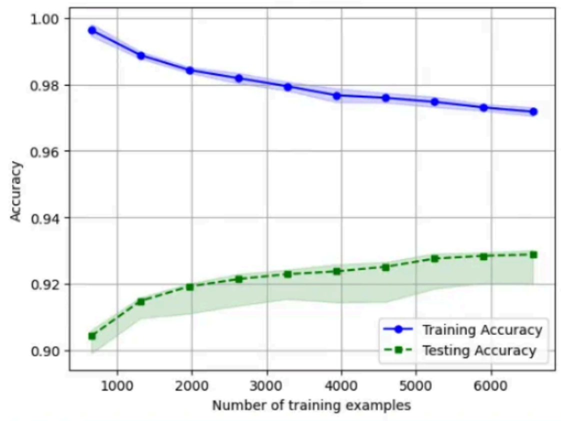
Insights:
From the feature importance and SHAP charts from 3 different models, we will be removing the features that are not important and do not contribute much to the prediction across the 3 models. These features are "State", "Policy", "Sales Channel", "Renew Offer Type", "Policy Type", reducing the dataframe from 30 columns to 18 columns.

Furthermore, our findings from the EDA section reinforce this conclusion, as the feature distributions for both classes are identical across "State", "Policy", "Sales Channel", and "Policy Type." Additionally, within each category of "Policy", the distributions among the features remain consistent. This suggests that there are minimal patterns to differentiate between the classes, which hampers effective classification.

Result:

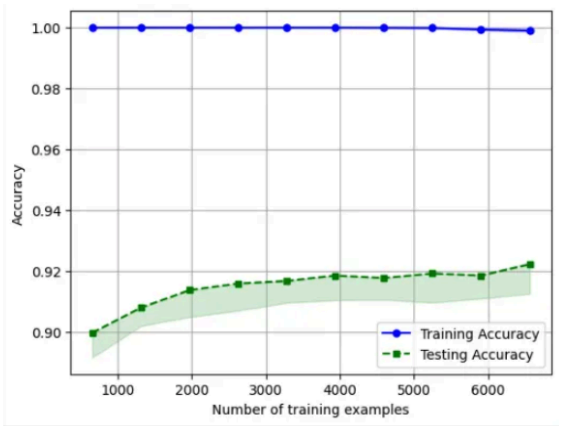
Algorithm \ Metric	Recall	F1 Score	ROC-AUC	PR-AUC
Light GBM	0.5228	0.6196	0.9116	0.9274
CatBoost	0.5275	0.6266	0.9176	0.9288
XG Boost	0.5226	0.6027	0.9005	0.9222

CatBoost:



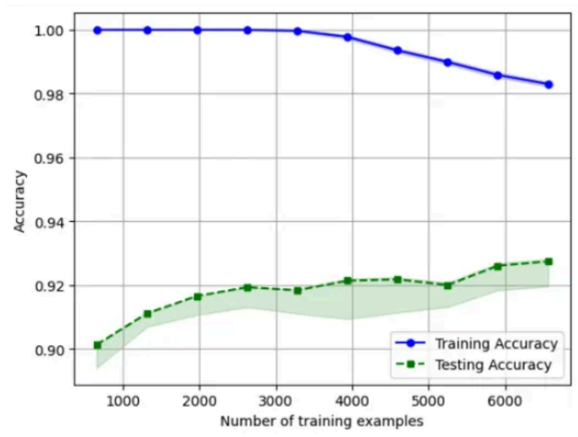
Confusion Matrix	Predict 0	Predict 1
Actual 0	635	11
Actual 1	35	48

XGBoost:



Confusion Matrix	Predict 0	Predict 1
Actual 0	629	17
Actual 1	34	49

LGBM:



Confusion Matrix	Predict 0	Predict 1
Actual 0	632	14
Actual 1	37	46

2.4.4 Hyperparameter Tuning

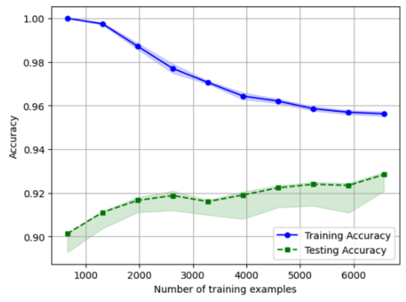
Machine learning models require pre-specified parameters; otherwise, default values are used, which may not yield the optimal model. To improve performance, we test a range of values for key parameters, including each model’s learning rate, maximum tree depth, number of trees, and L1 and L2 regularization strength. Using a grid search with 10-fold stratified cross-validation, we identify the best combination of these parameters for each model, maximizing predictive accuracy and ensuring balanced performance across classes.

Why model stacking?
Model stacking is used to boost predictive accuracy by combining multiple models, leveraging their unique strengths. Each model, or "base learner," captures different aspects of the data, and by stacking, we use a "meta-model" to learn the best way to combine their predictions. This layered approach helps reduce individual model biases, balances errors, and improves overall performance, often yielding results that outperform any single model alone.

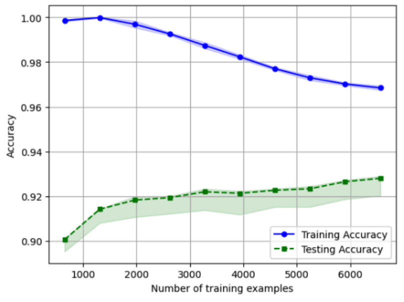
To build a strong stacked model with XGBoost, CatBoost, and LGBM, we aim to leverage the unique strengths each model offers rather than focusing solely on their individual weaknesses. Each algorithm will be tuned to excel in different areas to ensure that, when stacked, they contribute complementary perspectives, enhancing overall model robustness. Since XGBoost initially showed lower F1 scores, we focus on improving this metric to enhance its precision-recall balance. For CatBoost, we maximize accuracy, ensuring reliable base predictions. LGBM, known for capturing complex patterns, is also optimized for F1 to strengthen its classification balance.

After tuning, we achieve best F1 scores of **0.6288** for XGBoost, **0.6244** for LGBM, and an accuracy of **0.9288** for CatBoost, creating a well-rounded ensemble model.

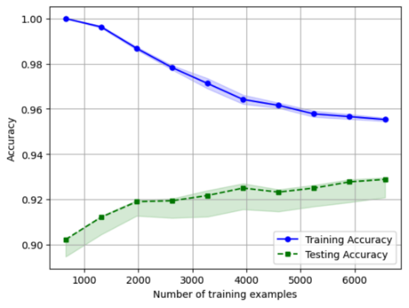
XGBoost:



LGBM:

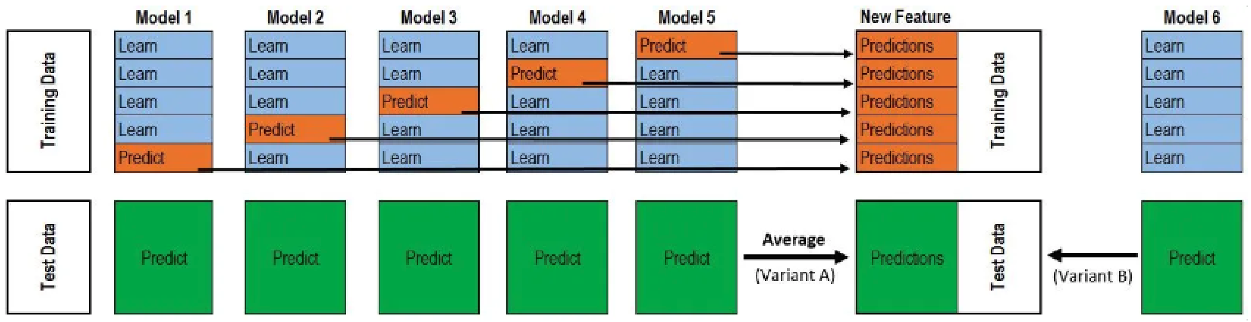


CatBoost:



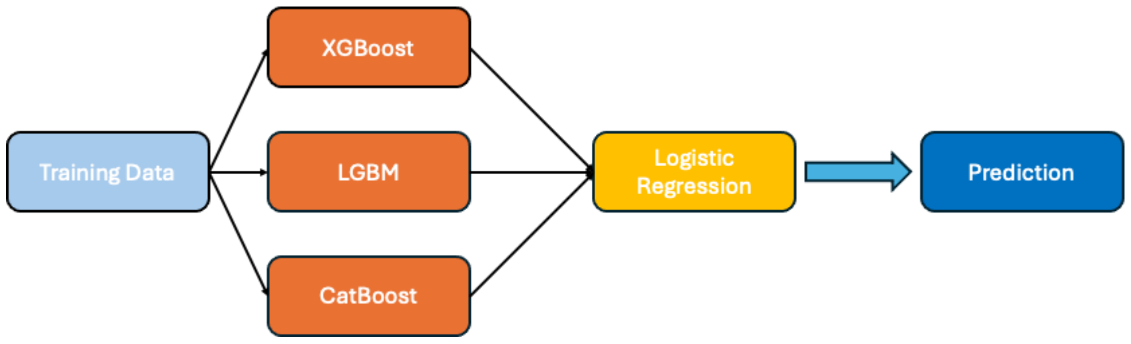
2.4.5 Model Ensembling

Model Stacking Specifics:

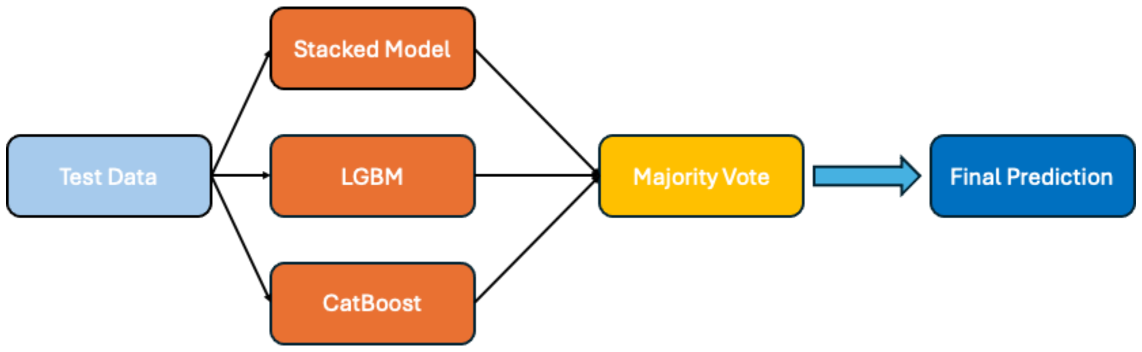


We use variant A, this stacking ensemble method with k-fold cross-validation, where multiple base models first generate predictions using a rotating fold system, each model learns on k-1 folds and predicts on the remaining fold. These out-of-fold predictions create a new feature matrix, where each column represents the averaged predictions from one base model type. The final meta-model (Model 6) is then trained on this newly created feature matrix along with the original target variable, effectively learning how to optimally combine the predictions from all base models. During prediction on new data, predictions from all folds of each base model are averaged to create the meta-features for the final model's prediction.

In our stacking setup, base models at the first level (XGBoost, LGBM, and CatBoost) generate initial predictions, which are then used as inputs to a meta-model (Logistic Regression) that learns how to best combine these outputs for the final prediction. This approach leverages the strengths of each model to improve overall predictive performance and reduce overfitting.



The stacked model's predictions in conjunction with LGBM (default parameter) and CatBoost (default parameter) for a majority vote ensemble provides an additional layer of robustness to the final prediction. We then incorporated LGBM and CatBoost using majority vote, both of which yielded good results in my previous submissions on the leaderboard. This allows the ensemble to make more informed and reliable predictions, ultimately improving accuracy and reducing the risk of overfitting.



2.5.6 Methods Attempted:

To address the class imbalance in our dataset, we explored several techniques to improve model performance. First, we upsampled the minority class to increase its representation, aiming to help the model learn patterns from these instances more effectively. Additionally, we tried Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples for the minority class. We also adjusted the class weights in our model to give more importance to the minority class predictions, hoping to mitigate the bias toward the majority class.

We also tried threshold tuning based on the Receiver Operating Characteristic (ROC) curve to achieve a better balance between sensitivity and specificity. However, these efforts did not significantly improve test set performance, as the model struggled to generalize from validation to test data, with leaderboard scores reflecting similar limitations. These results suggest that standard imbalance correction techniques were not effective in capturing our dataset's complexities, highlighting the need to explore alternative solutions.

3. Model Validation

Algorithm \ Metric	Recall	F1 Score	ROC-AUC	PR-AUC
CatBoost (Default parameters)	0.5275	0.6266	0.7540	0.9288
Light GBM (Default parameters)	0.5096	0.6048	0.7437	0.9244
Stacked Model	0.5144	0.6247	0.7490	0.9300
Ensemble Model	0.5204	0.6247	0.7511	0.9292

Ensemble Model's Confusion matrix:

Confusion Matrix	Predict 0	Predict 1
Actual 0	635	11
Actual 1	33	50

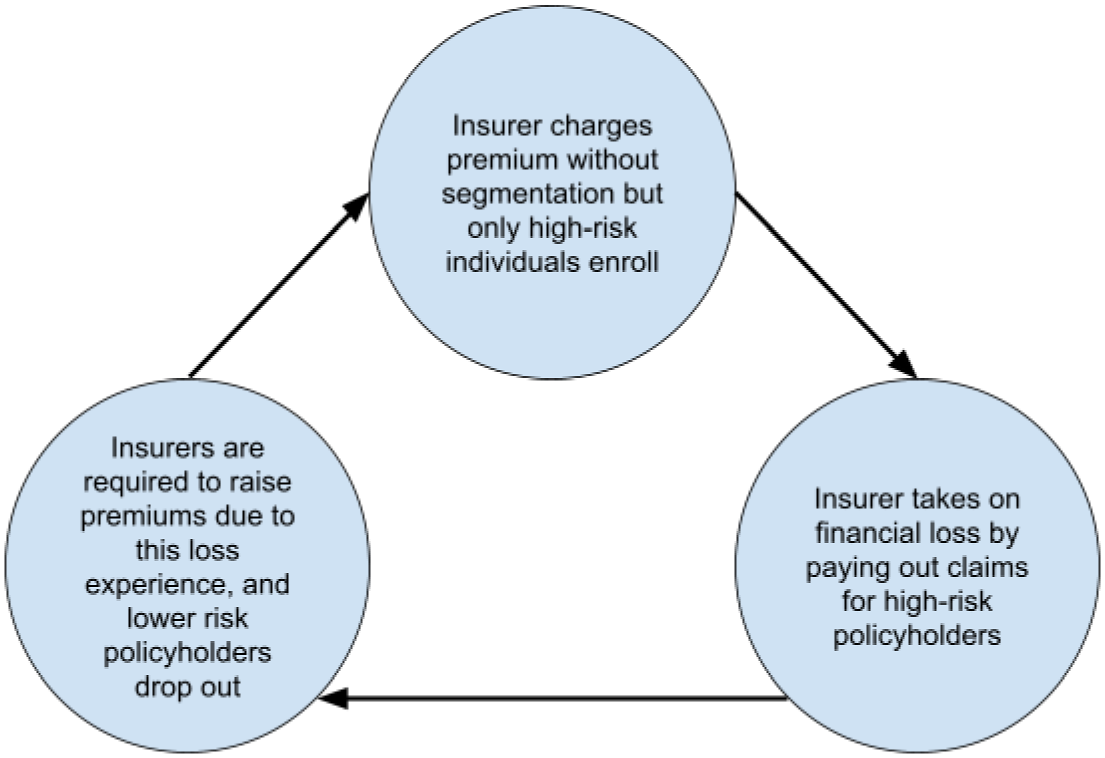
Based on the performance metrics shown in the table, the Ensemble Model demonstrates strong overall performance with the second highest accuracy (0.9292) among all models tested, including LGBM, CatBoost Model, and Stacked Model. The Ensemble Model achieved an F1 Score of 0.6247, Recall of 0.5204, and ROC AUC of 0.7511, indicating a good balance between precision and recall. While the Stacking Model had slightly higher accuracy (0.9300), the Ensemble Model's consistent performance across all metrics makes it a reliable choice for your final implementation. The relatively high accuracy and ROC AUC values suggest that the model is effective at distinguishing between classes.

4. Impact on business

4.1. Pricing Segmentation

The primary objective of insurance pricing is to segment risk and avoid adverse selection. Adverse selection is a concept where individuals who have high risk of making a claim are more likely to purchase insurance because they are more in need of coverage. On the other hand, lower risk individuals are less likely to purchase insurance, especially at a higher premium than what they are anticipating.

If a company enters the market without appropriate pricing, they are more likely to write higher risk individuals through offering them lower premiums than a suitable level, and lower risk individuals are harder to retain due to the need of raising premiums. This leads to a negative feedback loop, as shown below:


Insurer charges premium without segmentation but only high-risk individuals enroll
Insurer takes on financial loss by paying out claims for high-risk policyholders
Insurers are required to raise premiums due to this loss experience, and lower risk policyholders drop out

Through adequate pricing, an insurer is able to minimize their book of high risk policyholders, as well as growing their profitable business through lower risk individuals.

Through utilizing vast amounts of available policy data in conjunction with ML models, further segmented pricing structures can be achieved, leading to higher profitability for insurers, as well as more affordable insurance for the lower risk policyholders. Lower premiums for this subset of individuals can make the insurer more competitive in the industry, also leading to increased customer retention rate.

4.2. Underwriting Performance

In addition to the pricing of insurance products, insurers can also directly leverage the ML model in the underwriting aspect of the business.

Underwriting is essentially a pre-examination that insurers carry out to individuals attempting to buy insurance. Here, underwriters examine many different factors such as age, gender, education, and any other information that is useful to assess the risk level of an individual. This contributes to the pricing of the insurance policy as mentioned in the last section, but also determines whether an individual is eligible for insurance at all. If a person is deemed too risky for an insurer, they could be declined coverage

In an ideal world, insurance companies would only like to offer coverage to individuals with a moderate to low risk level, as a high risk policyholder would be too costly to insure. In reality, the accuracy of determining the risk level purely based on a list of variables is limited, and insurance companies still end up writing a handful of high risk individuals. Using the ML model proposed in this report would increase the accuracy of the classification of risk for individuals, and therefore increase profitability for an insurance company.

Here, we will demonstrate the expected increase in profitability using our model validation results and some simple assumptions. Below are the high-level results of our ensemble model mentioned above, listing the average accuracy, F1 score, ROC-AUC, and recall score of the model, followed by the confusion matrix. The metrics imply that the model has a balanced performance across various metrics with high accuracy ~94% and a good F1 score of ~0.70 suggesting a moderate balance between precision and recall. The confusion matrix is based on our testing, and the vertical values represent the 'True value' of classification, and the horizontal values represent the classification that the model outputs. As you can see, the model holds a ~99% accuracy for claims <$1000, and ~60% accuracy for claims >$1000.

Model is evaluated on an 80:20 split for training and test set.

Algorithm \ Metric	Recall	F1 Score	ROC-AUC	PR-AUC
Ensemble Model	0.6024	0.6969	0.7931	0.9403

Confusion Matrix	Predict 0	Predict 1
Actual 0	1271	21
Actual 1	66	100

Now, let us assume that the average premium for each policy is $500 a year, and we use our ML model to underwrite policies. Without using the ML model, we would have insured 166 policyholders that would go on to submit claims exceeding $1000, and the net loss for this subset of the book would be at least $1000 x 166 - $500 x 166 = $83,000.
In contrast, suppose we use our ML model and do not write any of the individuals that the ML model outputs as risky (Claim >$1000).
Then, our new net loss would be $1000 x 66 - $500 x 66 = $33000. This means that even using these conservative assumptions, we save $83,000 - $33,000 = $50,000 in losses. Note that this is only a dataset of ~1500 policies. With a larger book of business, the loss mitigated would be much more significant, and would drastically improve the profitability of the insurer.

4.3. Trend Selection

Finally, ML models can provide valuable insights to insurers about recent trends in the insurance market. Compared to using traditional methods such as linear regression, ML models will be able to consider a wide range of attributes, explore more complex data patterns, and reflect this into predicted future loss trends. Also, using ML models will allow insurers to react faster to recent trend movements in the industry, as the models will be responsive to even the slightest foreshadowing of future trends. Combining this with actuarial insight, insurance companies will be able to mitigate the risk due to changing industry trends.

4.4. KPIs

Despite the limitations in our dataset due to the lack of features to directly compute the necessary KPIs related to business, we can still make use of CLV, Months Since Last Claim (MSLC), and Months Since Policy Inception (MSPI) to derive a meaningful KPI. Below is a brief explanation on each variable:

- **CLV**: Higher CLV indicates more valuable customers. However, we won't be able to tell the behavior trend of such customers just from CLV. Thus, ratios as KPI are preferred as they are easier to act on.
- **CLV / MSLC**: This is a metric incorporating both the CLV and the risk factor of a customer (MSLC). Since there are cases where MSLC > MSPI, we assumed this would just be that the policyholder had an accident with the prior carrier. We focused on capturing the cases where MSLC <= MSPI.
    - This ratio gives a per-month revenue estimate since the customer's last claim. A higher ratio means that for each month without a claim, the customer brings in more revenue, as those with lower ratio might indicate a customer who has recently filed claims, hence costing the company more in payouts.

We want to capture such ratios from customers with correct classification. Out of 1458 records in our prediction test set, we correctly classified 1371 customers. By taking the cumulative sum of those ratios, we could get an estimate how much cumulative revenue we are bringing in relative to how long it's been since customers last made a claim.

5. Concerns and Future Improvements

5.1. Concerns

Although we are confident that the use of this ML model will provide valuable insights for insurers, there are a few concerns that should be addressed before the implementation of this model. We have briefly highlighted them below, along with our recommendation on how to overcome these.

Privacy
When training and testing the model we will use vast amounts of data collected from current policyholders. This often contains sensitive information, which is especially the case when training ML models that can take into account a large number of variables that contribute to the model. To keep compliant with privacy regulations and possible data breaches, only individuals who require it for their day-to-day work should have access to the data.

Fairness and bias
As ML models only take into account whether a variable is contributing to the model or not, it does not comprehend the meaning of the variable. This is especially problematic if a ML model is trained on a dataset with more variables, as it can leverage discriminatory variables that are not compliant with the regulator's guidelines. This can be overcome by frequent and adequate maintenance of the model, especially monitoring the variables that are contributing to it.

Explainability
The model is highly likely to become more complex as it utilizes more variables. This may be an issue when the model is being audited or submitted to regulators, as it is more difficult to explain the results clearly compared to a traditional method. To work around this, we can use SHAP values to see the overall importance of the variables, or use graphs to visualize individual trees within the model.

5.2. Possible Improvements

Due to limitations in time and resources, the developed ML model is far from completion. We have listed a few possible improvements that can be implemented in the future to improve the accuracy and practicality of the model.

Actual claim amount
Currently, the model that we have developed solves a classification problem, namely whether a claim will exceed $1000 or not. To make this model more practical, it would be preferable to convert this model into a regression problem, which would predict frequency or severity of claims.

More variables
One of the main strengths of a ML model is that it can take into account a large number of variables and construct an accurate prediction based on these. Currently, our dataset only includes less than a total of 30 variables, which does not maximise the utilization of this model. For improved accuracy, we should train the model on more variables.

Time trend analysis
Another future improvement is to add a variable for when the claim occurred, so we are able to analyse the underlying trend of claims over time. This could lead to many useful insights, and lead to better predictability of future loss trends.