Machine Learning Project Report:

# Auto Insurance Policy Lapse Risk Prediction

Author: Henry Udeogu => [Github Link](https://github.com/HenryUdeogu/insurance-policy-lapse-prediction_complete)

# Objective

This machine learning project uses a real-world, insurance-aligned dataset from the paper “***Dataset of an actual motor vehicle insurance portfolio by Segura-Gisbert et al”***. The authors of this paper conducted a research project within a Spanish insurance company and gained access to a sample of their motor vehicle insurance portfolio datasets which they were also authorized to share. This dataset is a collection of “105,555 records”, and the data has been anonymized to protect the policyholders' identities.

This dataset includes indispensable date-related information, including the effective date of policies, the birthdates of insured individuals, and renewal dates. It is also enriched with valuable economic variables, notably premiums and claim costs.

It is important to mention that the availability of open access data concerning insured populations is currently limited. This dataset can be used by insurance companies, researchers and educators and is relevant for marketing purposes; including customer segmentation, contract renewal processes, price renewal strategies, optimization and price sensitivity models, as well as pricing mechanisms for new business.

**The primary goal of this project is to build a classification model that predicts whether a customer is likely to let their auto insurance policy lapse (i.e., churn), so retention campaigns can be better targeted.**

Business Impact:

* **Improve Policyholder Retention**: Identify customers at risk of lapsing and implement proactive engagement strategies.
* **Optimize Marketing & Outreach**: Personalize communication strategies based on predicted lapse risk scores.
* **Reduce Revenue Losses:** Mitigate potential revenue decline due to policyholder churn.

# Data Source & Overview

As earlier mentioned, data was sourced from a research paper “***Dataset of an actual motor vehicle insurance portfolio by Segura-Gisbert et al”*** via this link <https://doi.org/10.17632/5cxyb5fp4f.2>.

The dataset is formatted as a spreadsheet covering the main operations of the company during a period of three (3) full years (November 2015 to December 2018), containing several motor insurance portfolio variables. This dataset comprises “**105,555 rows**” and “**32 columns**”. Each row signifies a policy transaction, while each column represents a distinct variable.

There are three (3) files in the data/raw folder:

* **Descriptive of the variables.xlsx** : Description of the variables in the dataset
* **Motor vehicle insurance data.csv** : Full motor insurance dataset (105K+ rows, 30 columns)
* **sample type claim.csv** : Partial claim type data (only 15% of policies, 2 additional columns relating to "claim\_type")

All dataset files have been included in the github repo for this project.

**Categorizing the Variables**

Based on the description of the raw variables, I have categorized them under 5 Features sets.

1. **Customer Demographics** - Describe the policyholder’s personal background, such as Age, Gender, Income Level etc.

2. **Policy Details** - Describes the structure and lifecycle of the insurance policy, including tenure, renewal dates, and claims history.

3. **Policy Behaviour/Engagement** - Describes policyholder's relationship history with the insurer. E.g. payment method, products held & lapse records.

4. **Financial Metrics** - Describes variables reflecting the economic value of the policy. This includes premiums paid and claim costs.

5. **Vehicle & Driving History** - Describes the technical and historical data about the insured vehicle and driving characteristics.

The dataset “**sample type claim.csv**” holds more granular information for a sample of policies. Unfortunately, the Insurance Company, according to the authors, provided only 15% of the claim type data because they considered the full set too sensitive, as it could expose the risk structure and proprietary pricing models.

Even with this constraint, I believe that including “claim types” and “cost by type” variables in my final feature set will help in determining the composition of some of the most significant risk variables while adding domain-specific richness to this insurance policy lapse model.

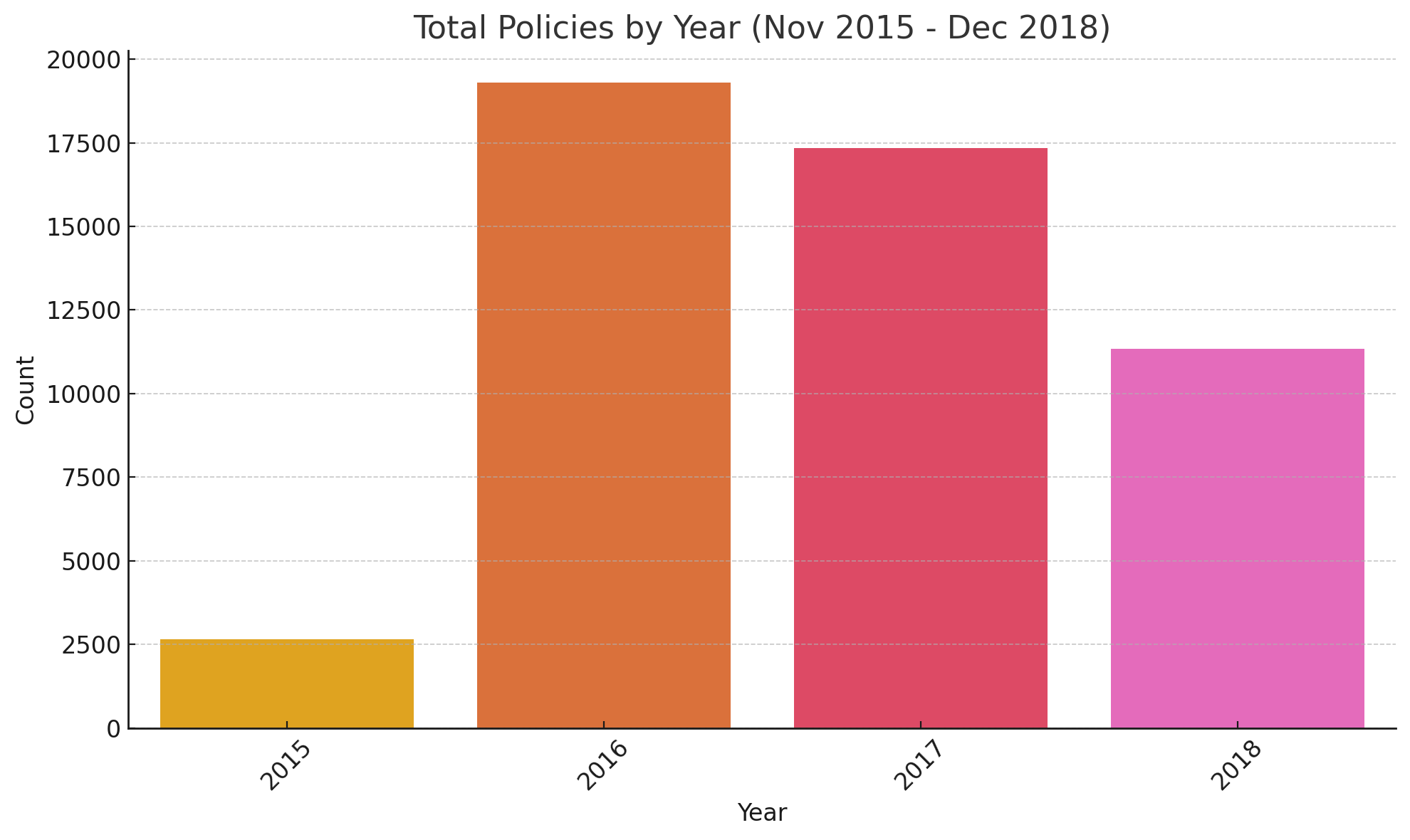
I will also add new features which will be engineered using the “**claim types**” and “**cost by type**”.

Below are some graphs that show the dataset distribution by year across certain groups between (Nov 2015 - Dec 2018).

1. Policy Lapse Distribution by Year
2. Total Policies by Year
3. Policy Distribution by Year and Age group

# 

# 

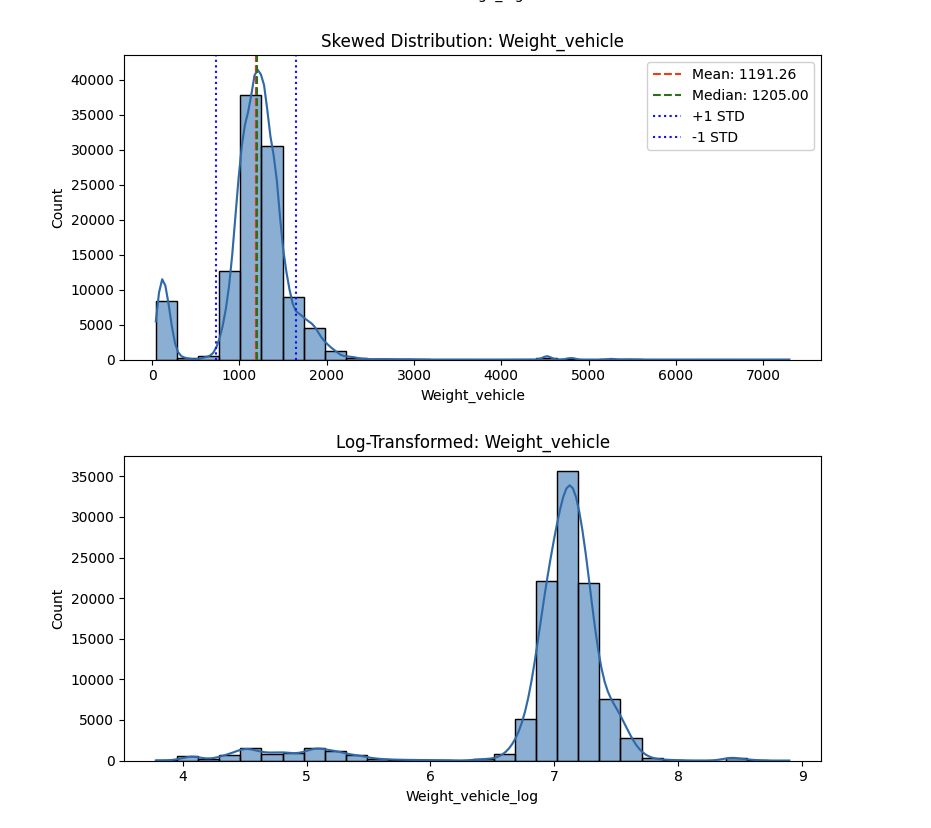


# 

# 

# Data Preprocessing

* Converted date columns into numerical features:
* Customer\_age from Date\_birth
* Tenure\_years from Date\_start\_contract to Date\_last\_renewal
* Generated visualizations (heatmaps, histograms, pandas profiling) to assess distributions and missingness.
* Handled missing data:
* Used **kNN imputation** for features FuelTypeDiesel and Length.
* Encoded categorical features using **one-hot encoding**
* Applied **log transformations** to normalize skewed numerical features (see below).



# 

# 

# 

# 

# Feature Engineering & Selection

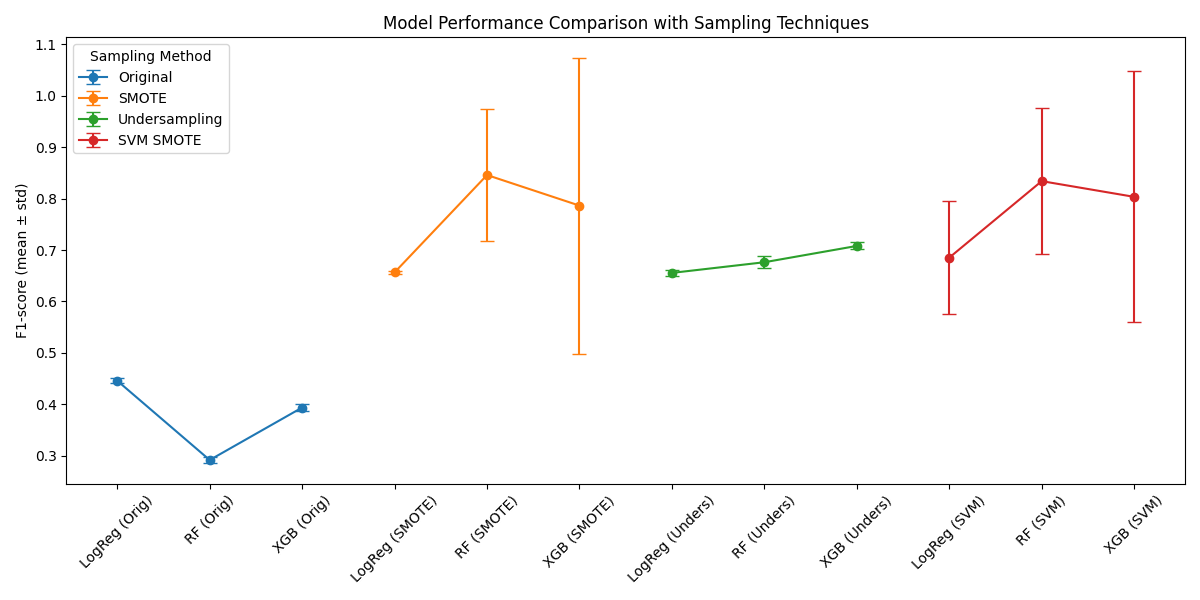
* Created features:
* HighSeverityClaimCost, Has\_serious\_claim, MinorClaimIndicator, DistinctClaimTypesFiled, Has\_fire\_claim etc.
* Selected features using:
* **Variance Inflation Factor (VIF)** to reduce multicollinearity
* Re-introduced some high-VIF features after SHAP analysis showed strong predictive power.
* Target Variable creation. 0 for Lapse < 1 in current year and 1 if Lapse >= 1

# Model Training & Selection

* Model was initially trained on a feature set including 44 features from the main dataset and claim type. 14 of which were engineered using the existing claims and policyholder information.
* Model was eventually trained on Top 20 features after SHAP feature importance analysis was done to identify the impact of each feature to the model prediction.
* Trained and compared multiple classifiers:
* Logistic Regression (with, without SMOTE, undersampling)
* Random Forest (with, without SMOTE, undersampling)
* XGBoost (with, without SMOTE , undersampling)
* Used **StratifiedKFold** cross-validation for robust evaluation
* Tuned classification **threshold** based on optimal F1-score from validation.
* Compared models using cross-validation F1-score and test set metrics.

| **Model** | **Logistic** | **Random Forest** | **XGBoost** |
| --- | --- | --- | --- |
| Original Data | 0.45 ± 0.005 | 0.29 ± 0.006 | 0.39 ± 0.007 |
| Undersampling | 0.66 ± 0.005 | 0.68 ± 0.011 | 0.71 ± 0.007 |
| SMOTE | 0.66 ± 0.002 | 0.85 ± 0.129 | 0.79 ± 0.288 |
| SVM SMOTE | 0.69 ± 0.110 | 0.83 ± 0.142 | 0.80 ± 0.244 |

Table 1: Summary of Results on Training data with Cross Validation (F1-score Mean ± Std)

Fig: Model Performance comparison (Training data) with sampling technique

Best Model Overall:

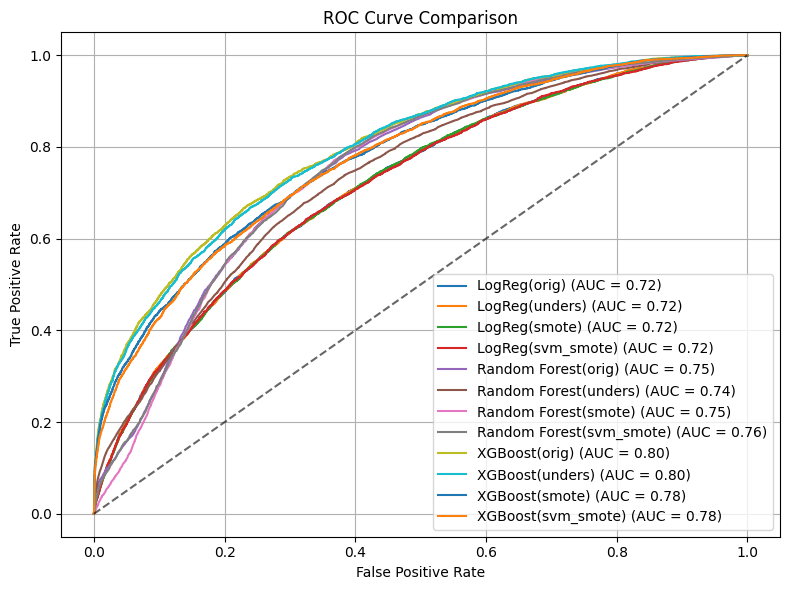
* XGBoost with Undersampling (F1 = 0.71 ± 0.007)
* Has the **highest training F1-score** (0.71) with low variance
* **Generalizes well** with a **test F1 of 0.51** and **ROC AUC of 0.80**
* Shows only a ~0.20 drop from train to test (expected for real-world problems)
* Outperforms all other models in **every evaluation metric**

Fig: ROC AUC plot for performance of each model and sampling technique on Test data

## Hyperparameter Tuning

* Tuned hyperparameters using **GridSearchCV** to find best params:
* param\_grid = {
  + 'n\_estimators': [100, 200],
  + 'max\_depth': [3, 5, 7],
  + 'learning\_rate': [0.01, 0.1],
  + 'subsample': [0.8, 1.0],
  + 'colsample\_bytree': [0.8, 1.0],
  + 'reg\_alpha': [0, 0.1, 1], # L1 regularization
  + 'reg\_lambda': [1, 5, 10] # L2 regularization}
* Best Params:
  + Best XGBoost Params : {'colsample\_bytree': 0.8, 'learning\_rate': 0.1, 'max\_depth': 7, 'n\_estimators': 200, 'reg\_alpha': 1, 'reg\_lambda': 5, 'subsample': 1.0}
  + Best XGBoost Score (CV score (F1)) : 0.7100940175445845

# Evaluation Metrics

Evaluated the model tuned model performance using:

* **F1-score** (especially for positive class)
* **ROC AUC**, **PR AUC**

F1, ROC AUC and PR AUC were chosen as metrics because of the class imbalance that exists in this dataset (20% Class 1 & 71% Class 0)

Plotted the following to visualize the performance of the model on the training (cross validation) and Test data.

* ROC Curve with AUC overlay for all models
* Precision-Recall Curve with best threshold highlighted
* F1-score vs Threshold graph to analyze balance
* Confusion matrix

Tracked and interpreted trade-offs (e.g., when recall increased but precision dropped)

# Results Summary

Following extensive sampling, training, hyperparameter & threshold tuning strategies, the **XGBoost model** trained and balanced with **undersampling** showed the best generalization to unseen data.

**Cross-Validation Performance (Training Set):**

* **F1-score (mean):** 0.71
* **Standard deviation:** 0.007
* Indicates stable and consistent model performance across folds

**Before Tuning Hyperparameters:**

* **Precision:** 0.40 (class 1)
* **Recall:** 0.70 (class 1)
* **F1-score:** 0.51 (class 1)
* **ROC AUC:** 0.80

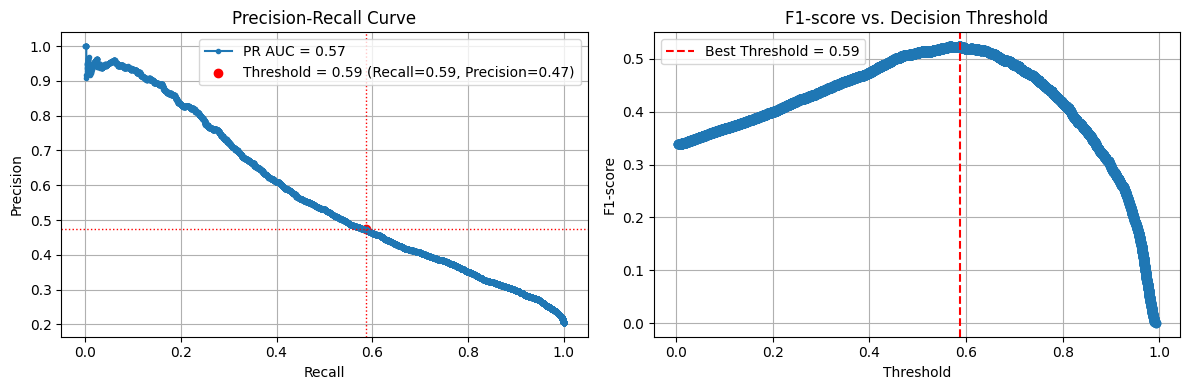
**After Tuning Hyperparameters:**

* **Precision:** 0.40 (class 1)
* **Recall:** 0.71 (class 1)
* **F1-score:** 0.51 (class 1)
* **ROC AUC:** 0.80
* **PR AUC**: 0.57 (Solid performance on minority class even with highly imbalance data)

**After Tuning Hyperparameters + Threshold (new threshold = 0.59):**

* **Precision:** 0.47 (class 1)
* **Recall:** 0.59 (class 1) (Recall score dropped but precision improved. More balance in f1-score)
* **F1-score:** 0.52 (class 1)
* **ROC AUC:** 0.80
* **PR AUC**: 0.57 (Solid performance on minority class even with highly imbalance data)

***Note***: In the case of Lapse risk identification, False Negatives are more costly as we want to be able to correctly identify when policyholders will lapse. More false negatives (lower recall) means more policyholders at risk of lapse will fall through the cracks. In this case, the default decision threshold (0.5) will be more beneficial.



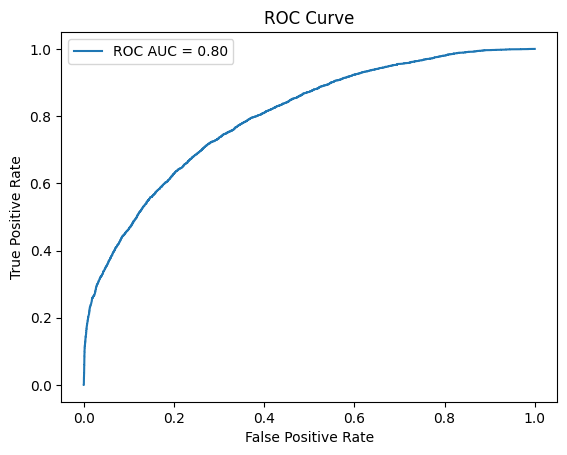


Fig: Shows performance of final model trained using the Top 20 features from SHAP

## 

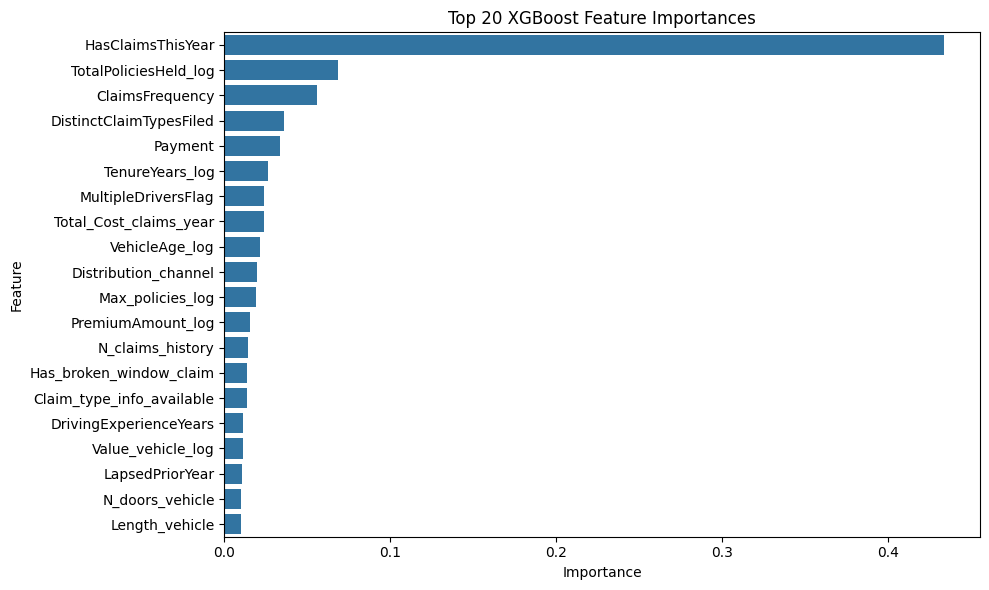
## 

## 

## Feature Importance

Feature importance shows **how much each feature contributes** to a model’s predictions. In this project, I have used both **XGBoost** feature importance and **SHAP**.

1. Feature Importance extracted using “model.feature\_importances\_”
2. Measures how often a feature is used in decision tree splits and how much it helps improve the model
3. Features with **higher scores** are considered **more useful** for making accurate predictions



## 

## 

## 

## 

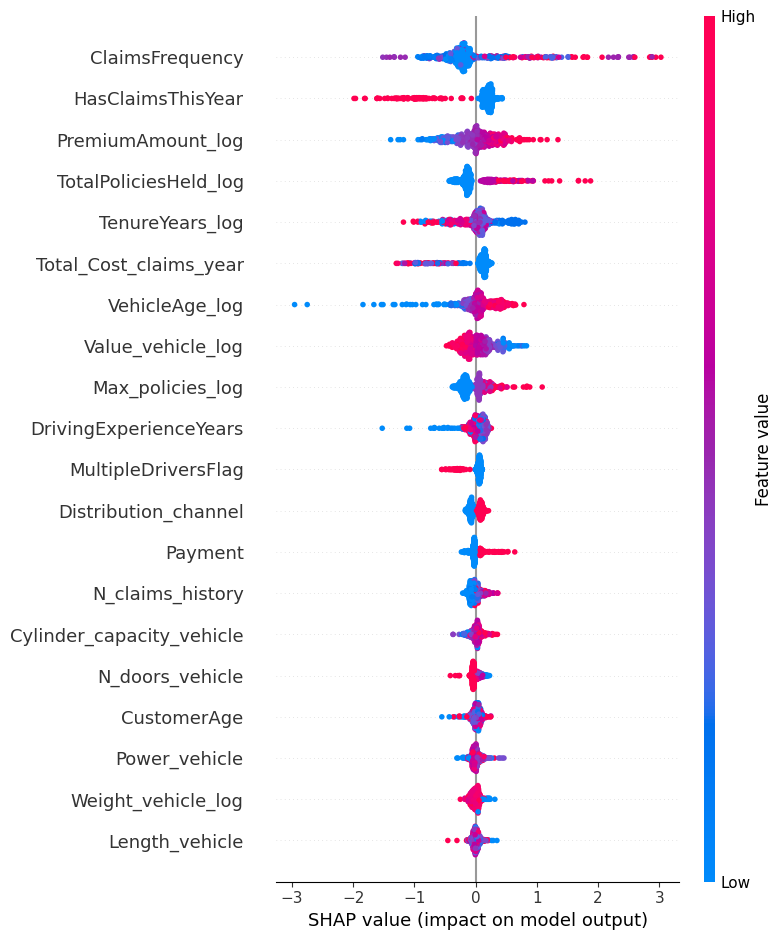
## 

## 

## SHAP

SHAP measures the **average absolute impact** of a feature on model predictions (across samples).

* It is based on game theory (Shapley values) — shows **how much each feature shifts the prediction**
* It is more trustworthy, especially in complex or imbalanced datasets
* It can also show **whether high/low values push predictions up or down**



# Model Explainability

Given that XGBoost is a collection of boosted gradient trees, interpretability can be challenging for XGBoost, like any complex model. Shapley Additive Explanations (SHAP) values are used to understand the impact of model variables on the dependent variable. To do this, the SHAP value is defined as the average contribution of a feature variable for all possible combinations of independent variables.

In summary, SHAP explains the **impact of each feature** on the model prediction.

* High SHAP value → strong **positive contribution**
* Low/negative SHAP value → **decreases** the prediction probability
* **SHAP summary plots** help visualize global trends across all predictions:  
  + Red → high feature value
  + Blue → low feature value

If most points for a feature are blue and far left → low values **decrease lapse risk** If red points cluster right → high values **increase lapse risk.**

**Top 20 Features (Final Feature Set)**

'ClaimsFrequency',

'HasClaimsThisYear',

'PremiumAmount\_log',

'TotalPoliciesHeld\_log',

'TenureYears\_log',

'Total\_Cost\_claims\_year', '

VehicleAge\_log',

'Value\_vehicle\_log',

'Max\_policies\_log',

'DrivingExperienceYears',

'MultipleDriversFlag',

'Distribution\_channel',

'Payment',

'N\_claims\_history',

'Cylinder\_capacity\_vehicle',

'N\_doors\_vehicle',

'CustomerAge',

'Power\_vehicle',

'Weight\_vehicle\_log',

'Length\_vehicle'

# Next Steps

This Lapse Risk Prediction Model is in no way at its best and can still be improved by adding more instances (training examples) of Lapse Risk to make the dataset more balanced without the need for under or over sampling. In addition to that, additional Customer Demographic, Telematics & Behaviourial type variables could be key in modeling Policyholders Lapse pattern

## Additional Variables Categories

### 1. Behavioral Variables

- Payment history (late payments, bounced charges)

- Days before renewal that payment was made

- Policy switching frequency, past lapse behavior

- Time since last interaction (support call, claim, etc.)

### 2. Engagement Metrics

- Response to marketing campaigns

- Call center/chat interaction frequency

- Mobile app or portal usage activity

### 3. Additional Demographic Variables

- Income

- Credit Score

- Gender

These additional variables could improve both prediction accuracy and explainability of the lapse model.