# Capstone Project - III

# HEALTH INSURANCE CROSS SELL PREDICTION

**BY**

**SHADAB MAHEMUD SHAIKH**

# Problem Definition

Our client is an Insurance company that has provided Health Insurance to its customers now they need your help in building a model to predict whether the policyholders (customers) from past year will also be interested in Vehicle Insurance provided by the company.

Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimise its business model and revenue.

# Business Understanding

**Cross Selling:**

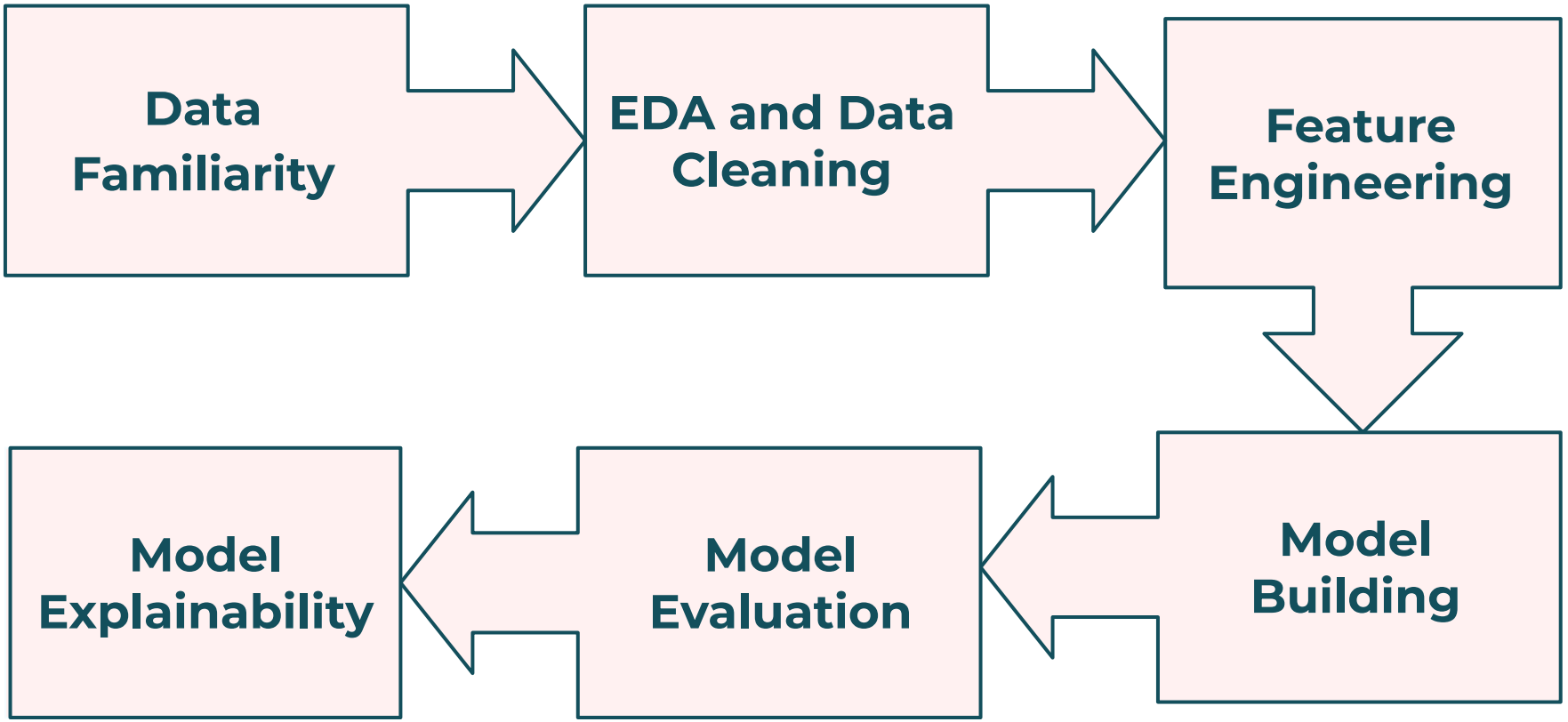
Cross-selling to existing clients has been one of the primary methods of generating new revenue for many businesses. Identifying the potential buyers among existing clients can help a company plan its communication strategy accordingly, thereby optimizing its business model and increasing revenue.

**What is an insurance policy?**

An insurance policy is an arrangement by which a company undertakes to provide a guarantee of compensation for specified loss, damage, illness, or death in return for the payment of a specified premium. A premium is a sum of money that the customer needs to pay regularly to an insurance company for this guarantee.

Just like medical insurance, there is vehicle insurance where every year customer needs to pay a premium of certain amount to insurance provider company so that in case of unfortunate accident by the vehicle, the insurance provider company will provide a compensation (called ‘sum assured’) to the customer.

# Project Pipeline



# Data Familiarity

# Let’s understand our dataset...

- **Dataset Name:** HEALTH INSURANCE CROSS SELL PREDICTION Dataset

- **Size and Shape:** The shape of dataset is (381109 x 12)
  i.e, 381109 observations and 12 features

- **Independent Variables:**
-
  - **Numerical Features:** ‘id’, ‘Age’, ‘Region_Code’, ‘Annual_Premium’, ‘PolicySalesChannel’, ‘Vintage’

  - **Categorical Features:** ‘Gender’, ‘Driving_License’, ‘Previously_Insured’, ‘Vehicle_Age’, ‘Vehicle_Damage’

- **Target Variable:** ‘Response’ (Dichotomous)

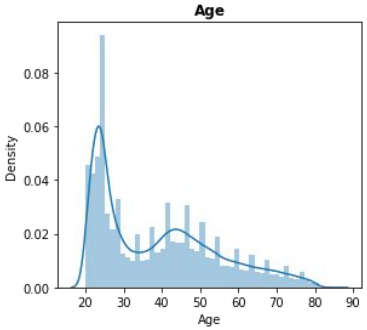
# Let’s understand our dataset...

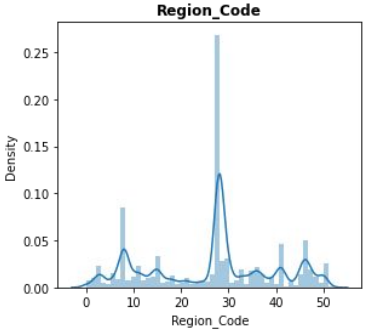
## Feature Information

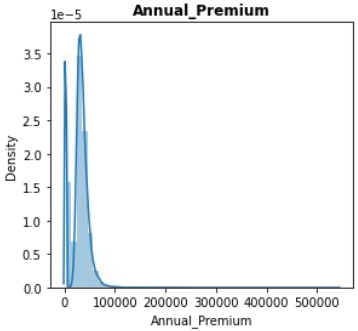
- **id:** Unique ID for the customer
- **Gender:** Gender of the customer
- **Age:** Age of the customer
- **Driving_License:** 0 : Customer does not have DL, 1 : Customer already has DL
- **Region_Code:** Unique code for the region of the customer
- **Previously_Insured:** 1 : Customer already has Vehicle Insurance, 0 : Customer doesn't have Vehicle Insurance
- **Vehicle_Age:** Age of the Vehicle
- **Vehicle_Damage:** 1 : Customer got his/her vehicle damaged in the past. 0 : Customer didn't get his/her vehicle damaged in the past.
- **Annual_Premium:** The amount customer needs to pay as premium in the year
- **PolicySalesChannel:** Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc.
- **Vintage:** Number of Days, Customer has been associated with the company
- **Response:** 1 : Customer is interested, 0 : Customer is not interested

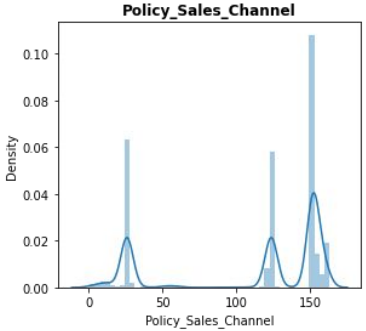
# EXPLORATORY DATA ANALYSIS
# AND
# DATA CLEANING

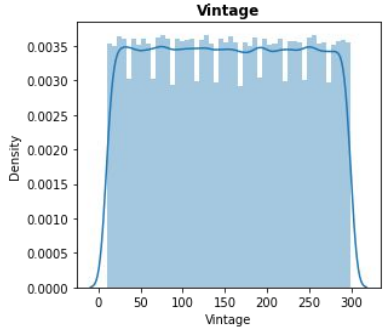
# Visualizing Distributions



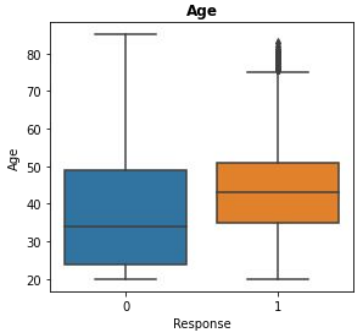



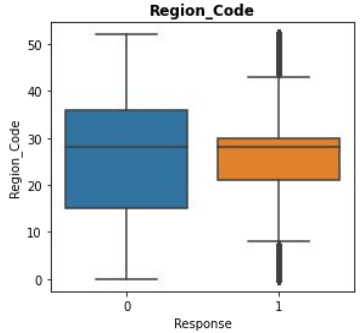



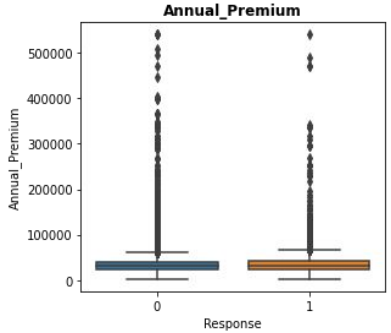



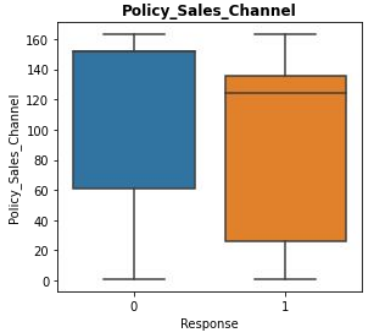



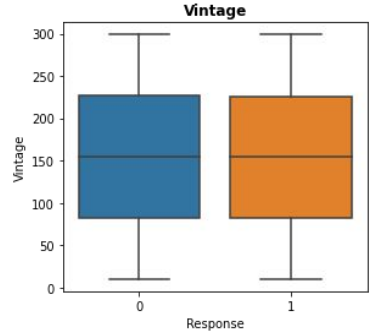

- Distribution of numeric variables suggest the features ‘Age’ and ‘Annual_Premium’ are skewed towards right.
- Box plots suggest that the ‘Annual_Premium’ consists of outlier observations



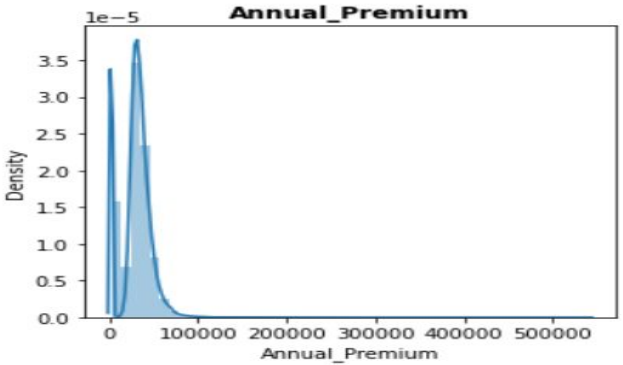



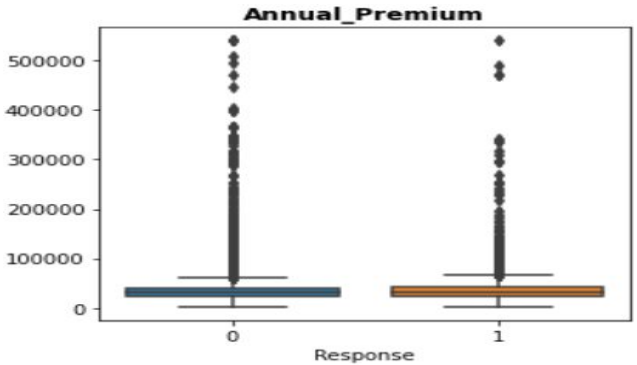



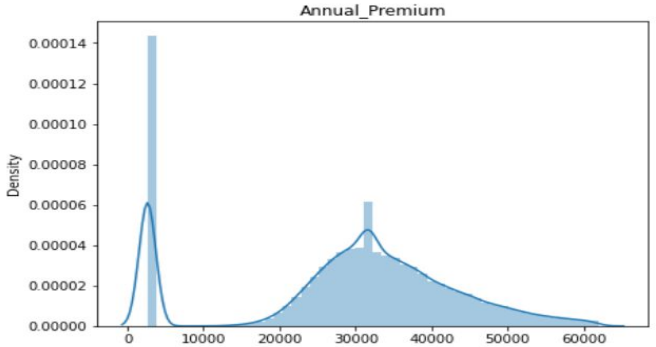
# Handling Outliers

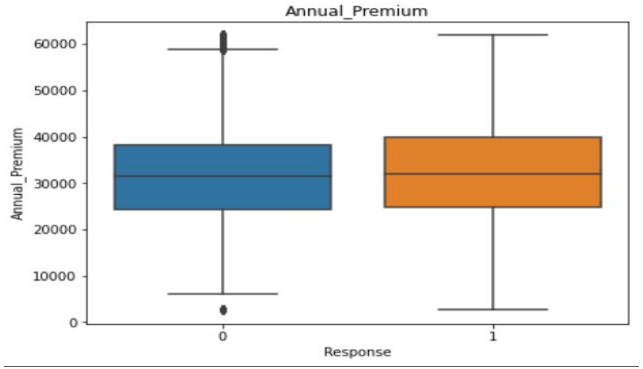
- We will implement **IQR** method to handle outliers in the 'Annual_Premium' Feature.
- Here, we replaced the outliers (observations that lie outside the ±1.5*IQR range) with its median value.



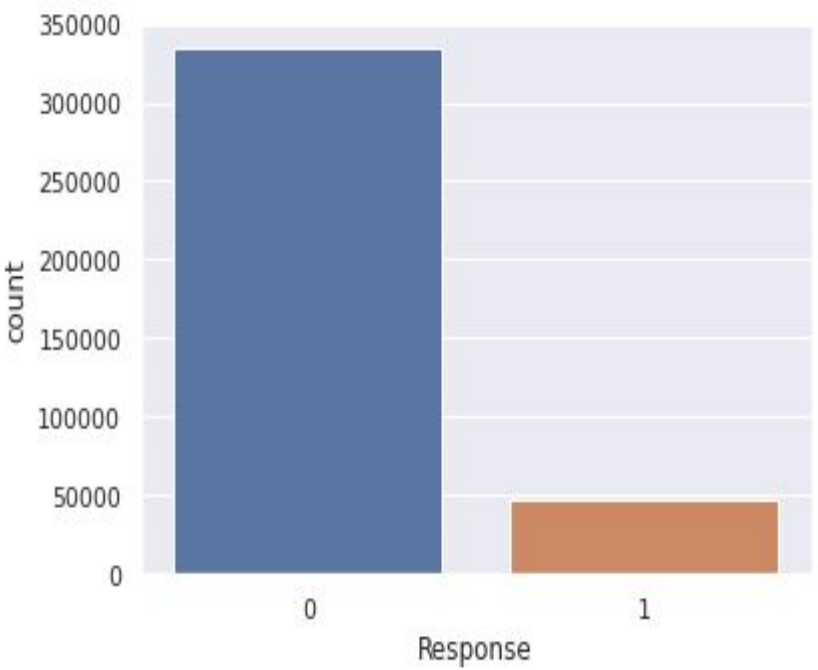

IQR



# Univariate Analysis

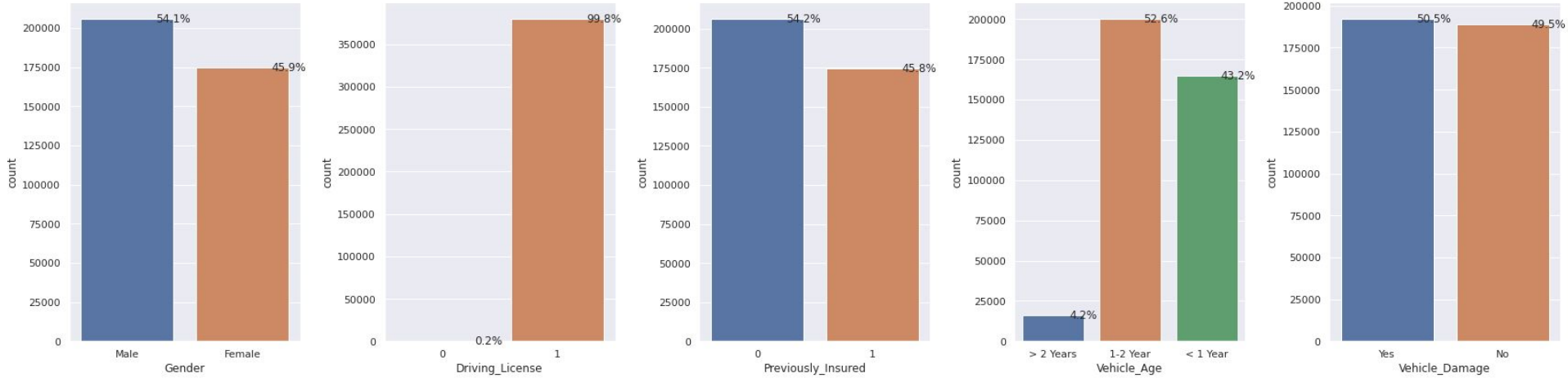
## Target Variable:




- Feature ‘Response’ consists of two labels;
  - 1 : Customer is interested,
  - 0 : Customer is not interested
- There is a heavy class imbalance as seen in the graph.
- There are 334399 or 87.74% **Class 0** observations.
- There are 46710 or 12.26% **Class 1** observations.

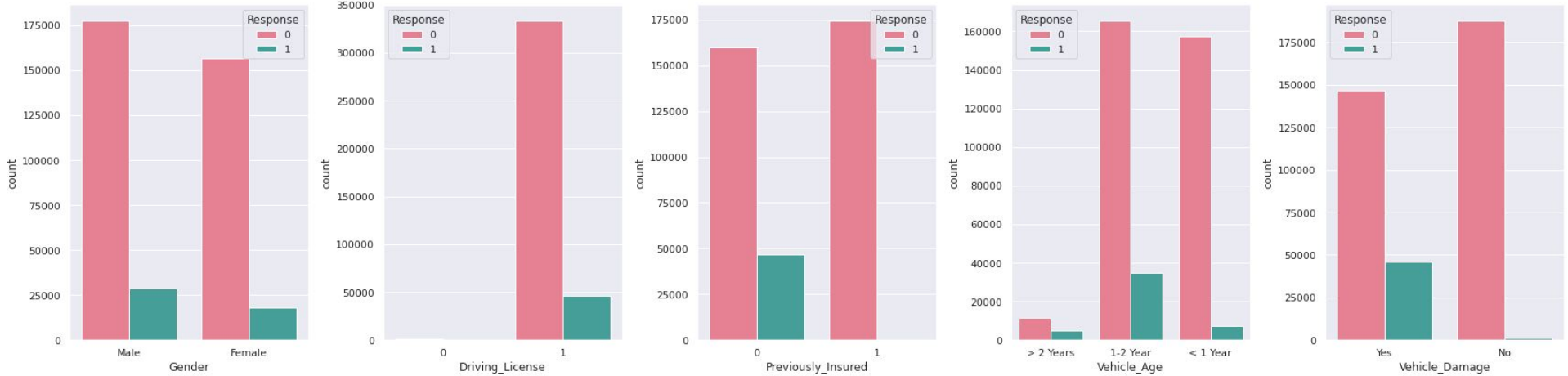
# Univariate Analysis

- Gender column suggests that number of Male customers(54.1%) is higher than Female customers(45.9%).
- Almost all of the customers possess a driving license.
- About 45.8% of the customers have been previously insured.
- Vehicles with age 1-2 years are prevalent in our data(52.6%).
- Almost half the vehicles have already suffered some damage(50.5%).



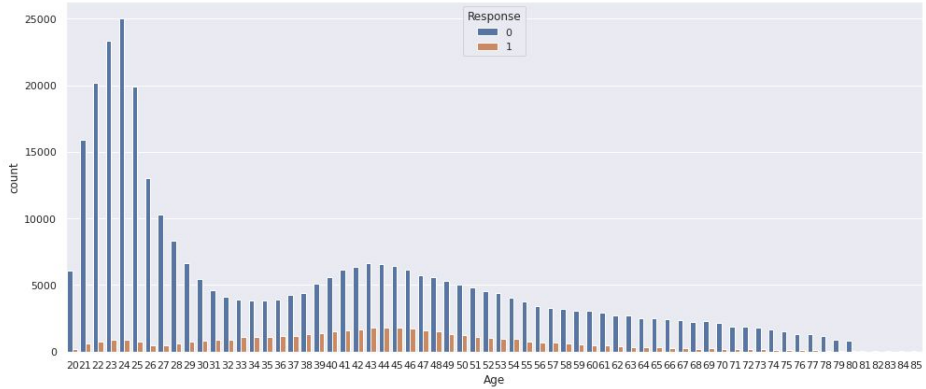
# Bivariate Analysis

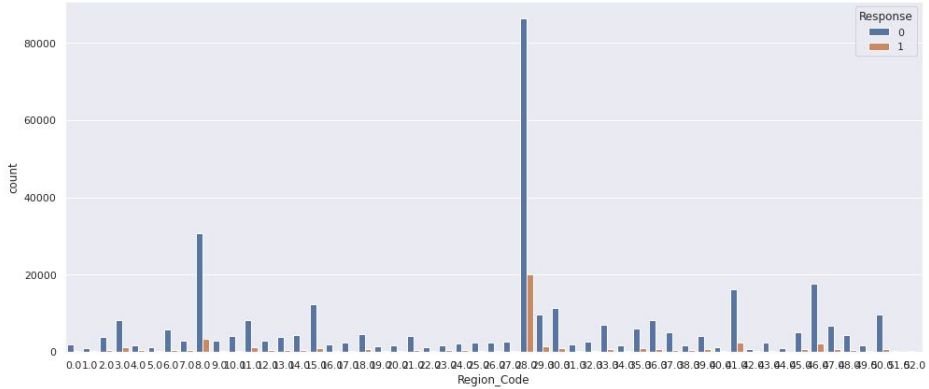
- The Response is similar from both Gender groups.
- Driving_Licence feature appears to be redundant as previously seen, most observations suggest that the user possesses a driving license.
- The customers who are previously insured do not show a positive response.
- Customers having 1-2 years old and already damaged vehicle seem to show a positive response.

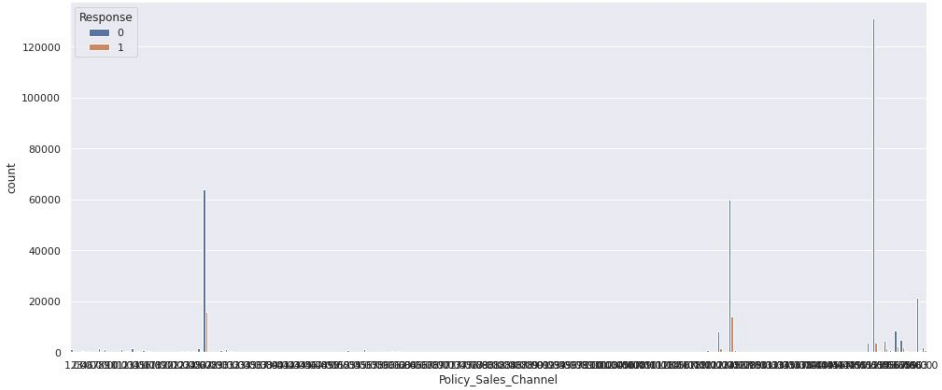


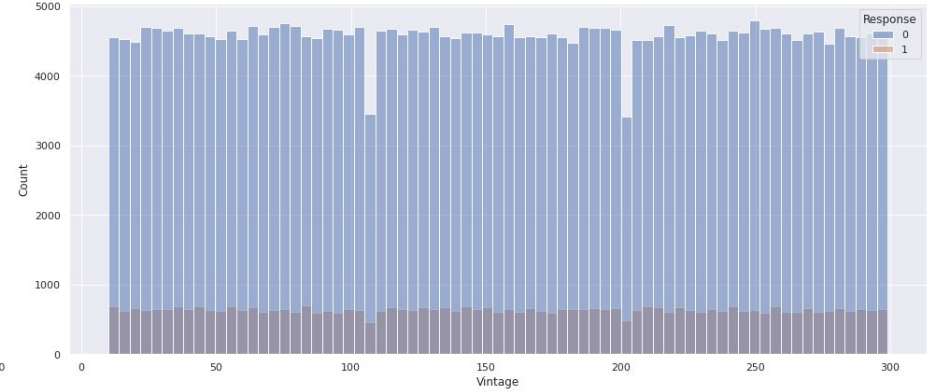
# Bivariate Analysis

- There is a huge dispersion of numeric features w.r.t the target variables.
- Data shows that the customers af age 35-55 years are most likely to buy the vehicle insurance.



# FEATURE ENGINEERING

# FEATURE ENGINEERING

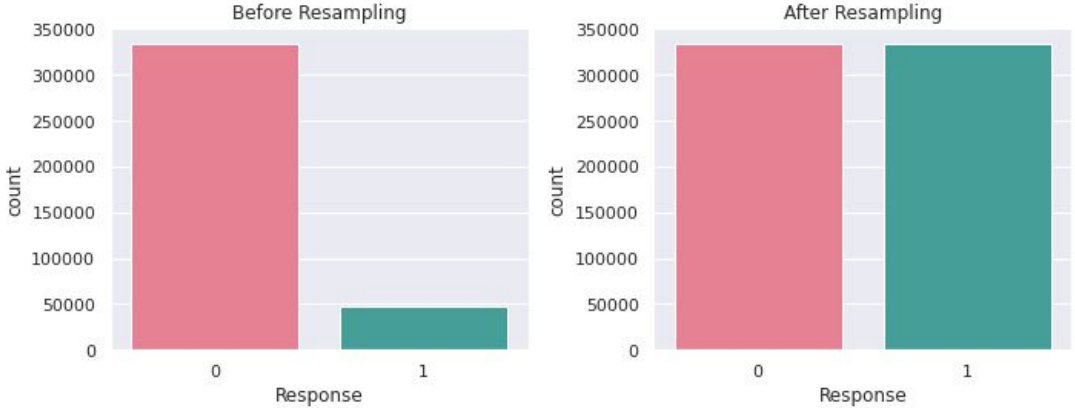
- **Created new features:**
  - Region: Binning the values from ‘Region_code’ column to get 5 distinct Region Classes
- **Label Encoding:**
  - Gender: 1 = Male, 0 = Female
  - Vehicle_Damage: 1 = Yes, 0 = No
- **One Hot Encoding:**
  - Vehicle_Age: New Features = ‘Vehicle_Age_1-2 Year’, ‘Vehicle_Age_< 1 Year’ and ‘Vehicle_Age_> 2 Years’
  - Region: New Features = ‘Region_A’, ‘Region_B’, ‘Region_C’, ‘Region_D’ and ‘Region_E’

# MODEL BUILDING
# AND
# MODEL EVALUATION

# Prerequisites

- **Independant and Target Variables:**
  - We put ‘Response’ feature into the independent variable ‘y’
  - We put the remainder original, cleaned and engineered (total 15) features into dependent variable ‘X’.

- **Class Imbalance Handling:**
  - Our data contains heavy class imbalance w.r.t the predictor label.
  - We implemented SMOTE to handle class imbalance and got the following results:




```
Before resampling:
 0    334399
1     46710
Name: Response, dtype: int64
After resampling:
 1    334399
0    334399
Name: Response, dtype: int64
```

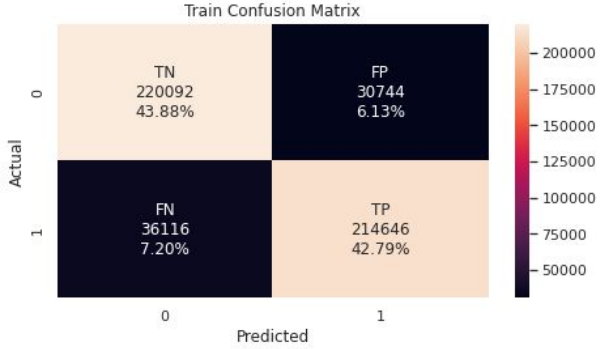
# Prerequisites

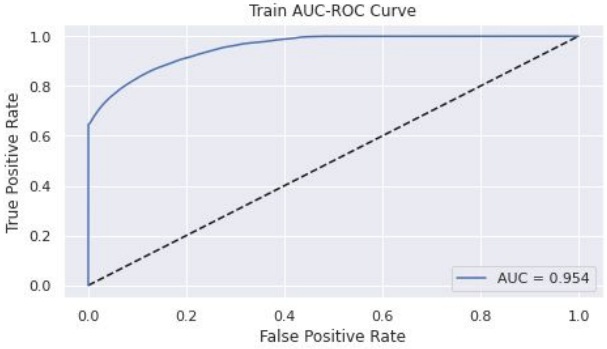
- **Train-Test Split:**
  - Performed train-test split using sklearn.
  - Training Data - 75% | Test Data - 25%

- **Feature Scaling:**
  - Normalized data using sklearn MinMaxScaler

- **Defining Helper function:**
  - Defined model_eval() function.
  - Inputs: model, X_train, X_test, y_train, y_test
  - Outputs
    - Model Training Time
    - Best Model Parameters(For tuned models)
    - Visualizations: Train and Test Confusion Matrix, AUC-ROC Curve
    - Train and Test Evaluation Metrics(Accuracy, Precision, Recall, F1 Score, Log Loss)

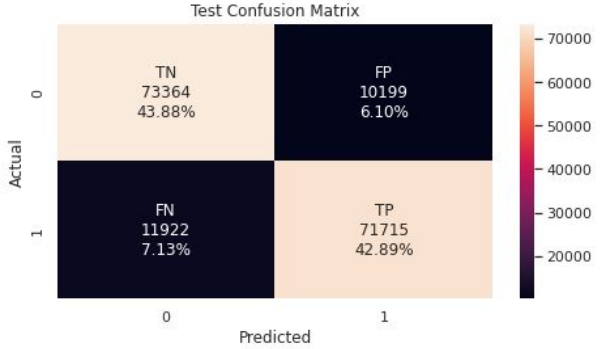
# Logistic Regression

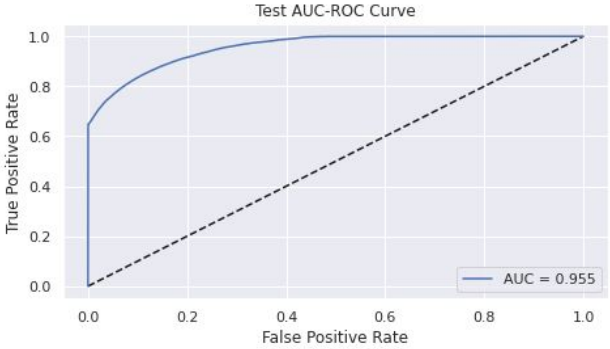



Train Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8667060075997113 |
| Precision | 0.8747137210155264 |
| Recall | 0.8559749882358572 |
| F1 Score | 0.8652429094309809 |
| ROC-AUC Score | 0.8667044247020552 |
| Log Loss | 4.603860407312473 |



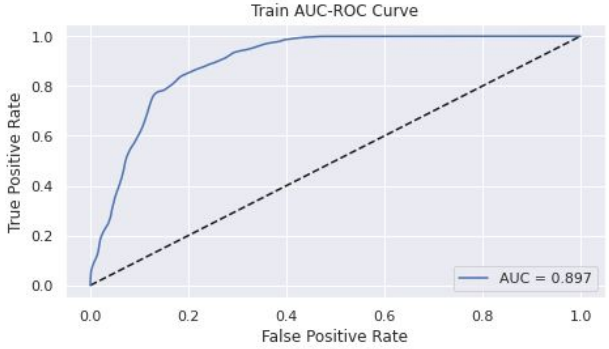




Test Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8676973684210526 |
| Precision | 0.8754913689967527 |
| Recall | 0.8574554324043187 |
| F1 Score | 0.8663795446720346 |
| ROC-AUC Score | 0.8677019033424008 |
| Log Loss | 4.569619783050299 |

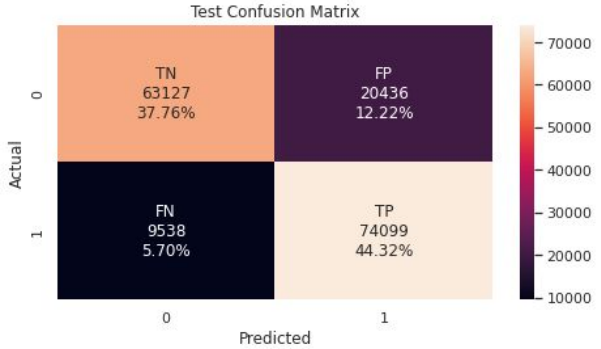
- The best parameters found out to be : {'solver': 'liblinear', 'random_state': 2, 'penalty': 'l2', 'C': 10}

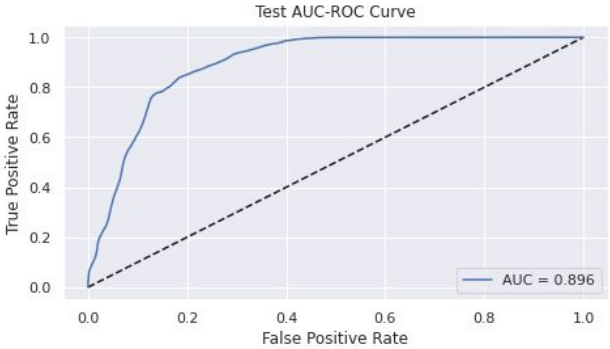
# Gaussian Naive Bayes


Train Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8220407577382685 |
| Precision | 0.7848349524152186 |
| Recall | 0.8872795718649557 |
| F1 Score | 0.8329190500434248 |
| ROC-AUC Score | 0.8220503809028968 |
| Log Loss | 6.146591712804587 |



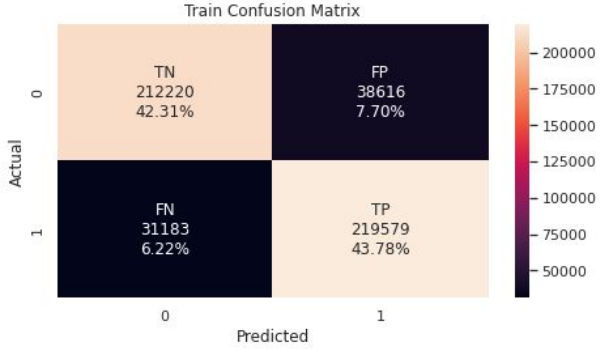



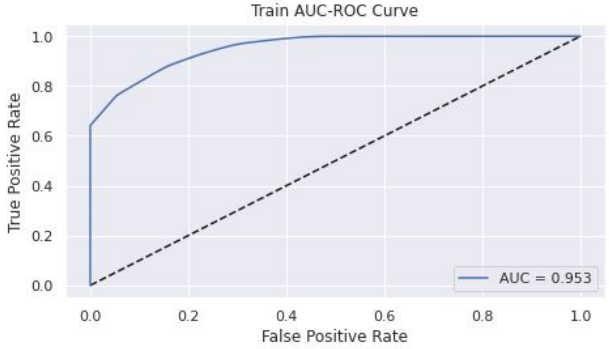

Test Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8207296650717704 |
| Precision | 0.7838260961548633 |
| Recall | 0.8859595633511484 |
| F1 Score | 0.8317693015737603 |
| ROC-AUC Score | 0.8207007825970346 |
| Log Loss | 6.19187574302713 |

- The best parameters found out to be : {'var_smoothing': 0.0028480358684358025}

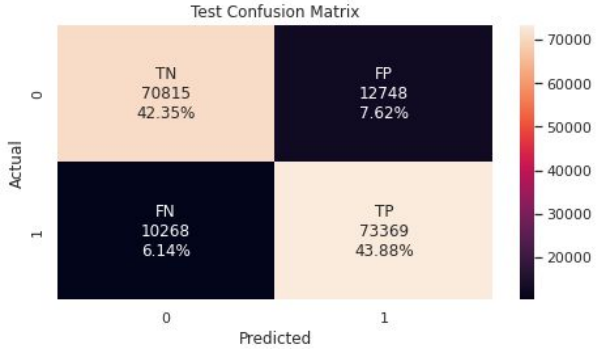
# Decision Trees

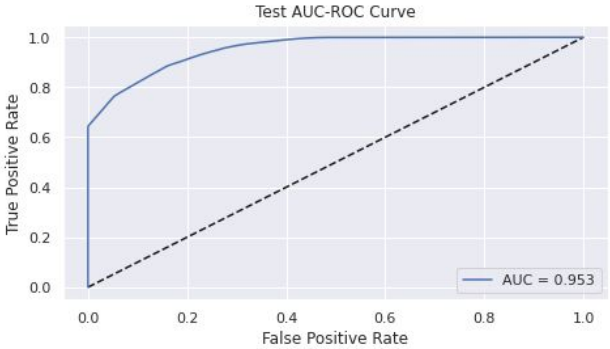



Train Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8608467338386516 |
| Precision | 0.8504386219717656 |
| Recall | 0.8756470278590855 |
| F1 Score | 0.8628587483814939 |
| ROC-AUC Score | 0.8608489169817363 |
| Log Loss | 4.806245102338397 |



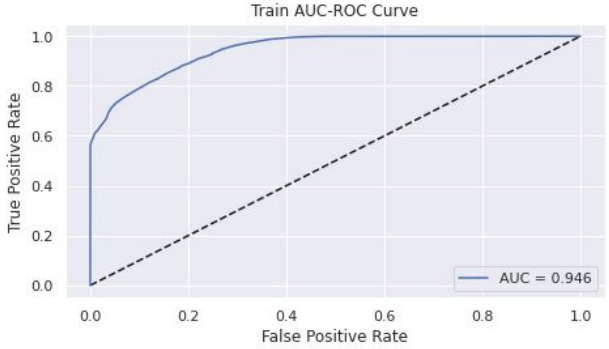




Test Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8623444976076555 |
| Precision | 0.8519688330991558 |
| Recall | 0.8772313688917585 |
| F1 Score | 0.8644155660544082 |
| ROC-AUC Score | 0.8623379060032671 |
| Log Loss | 4.754513581180051 |

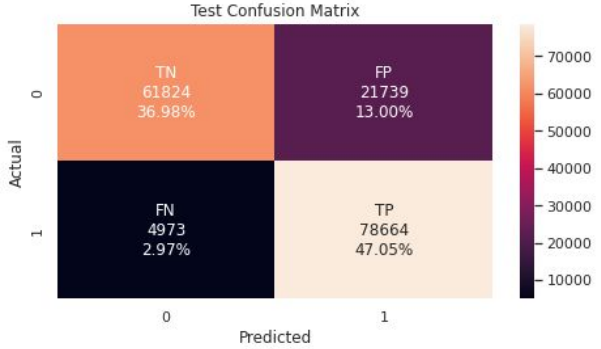
- The best parameters found out to be : {'splitter': 'random', 'random_state': 23, 'min_weight_fraction_leaf': 0.0, 'min_samples_leaf': 3, 'max_leaf_nodes': 60, 'max_features': None, 'max_depth': 9}

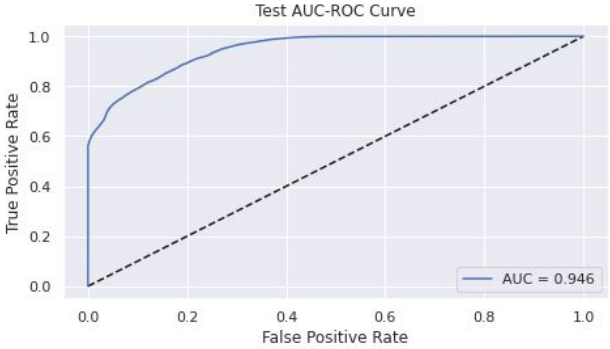
# Random Forest


Train Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.839686362385815 |
| Precision | 0.7827660273754312 |
| Recall | 0.9402700568666704 |
| F1 Score | 0.8543192609890558 |
| ROC-AUC Score | 0.8397011991584304 |
| Log Loss | 5.537141192552199 |



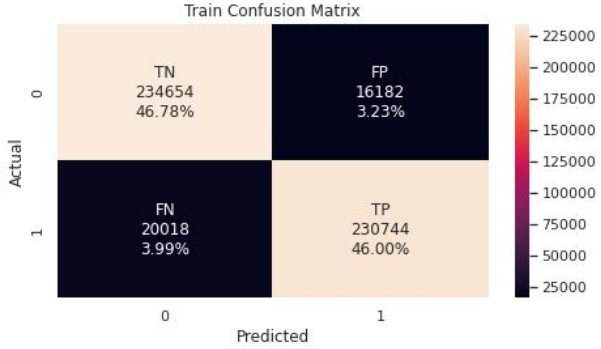



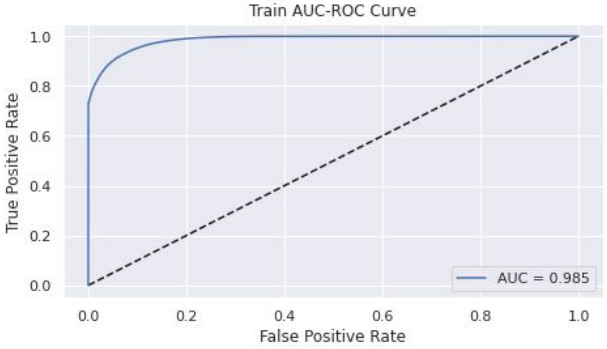

Test Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8402392344497608 |
| Precision | 0.7834825652619941 |
| Recall | 0.9405406697992515 |
| F1 Score | 0.8548576396435557 |
| ROC-AUC Score | 0.8401948230103925 |
| Log Loss | 5.518045320032232 |

- The best parameters found out to be : {'n_estimators': 150, 'max_depth': 7, 'criterion': 'entropy'}

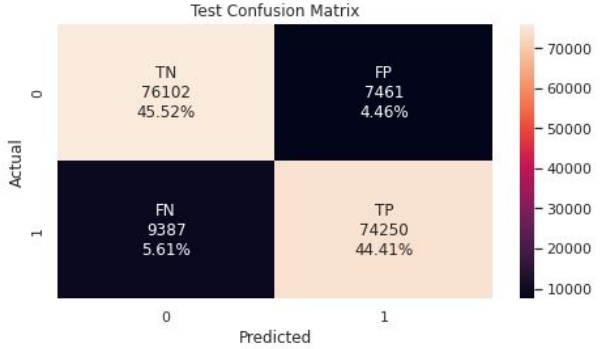
# XGBoost

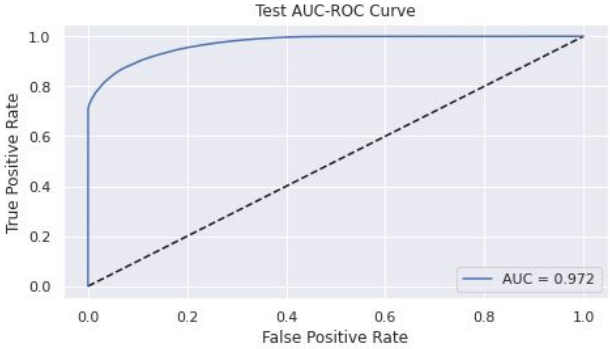



Train Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.9278306532322697 |
| Precision | 0.9344661963503236 |
| Recall | 0.9201713178232747 |
| F1 Score | 0.9272636671971194 |
| ROC-AUC Score | 0.9278295234286923 |
| Log Loss | 2.492666726305492 |



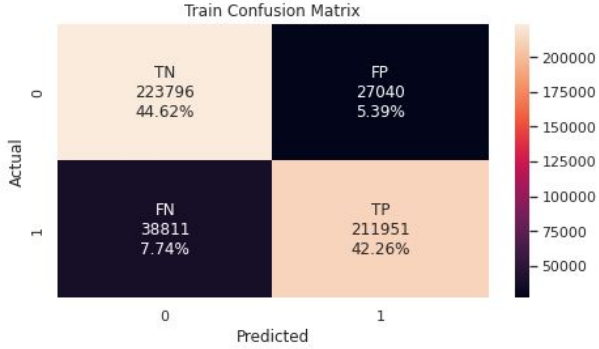



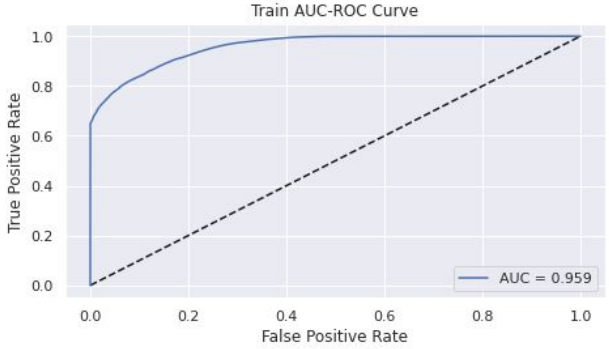

Test Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8992344497607655 |
| Precision | 0.9086903844035686 |
| Recall | 0.8877649843968578 |
| F1 Score | 0.8981058131939907 |
| ROC-AUC Score | 0.8992395282071888 |
| Log Loss | 3.480354488623696 |

- The best parameters found out to be : {'n_estimators': 100, 'max_depth': 15, 'criterion': 'entropy'}

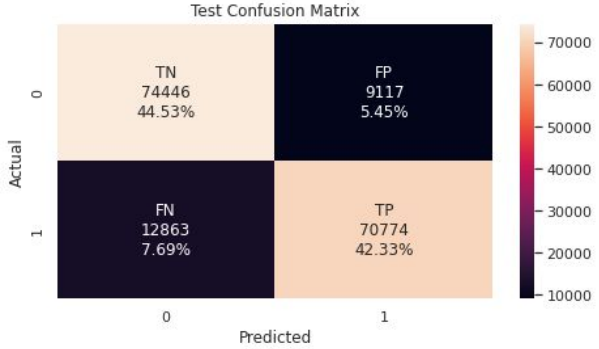
# LightGBM

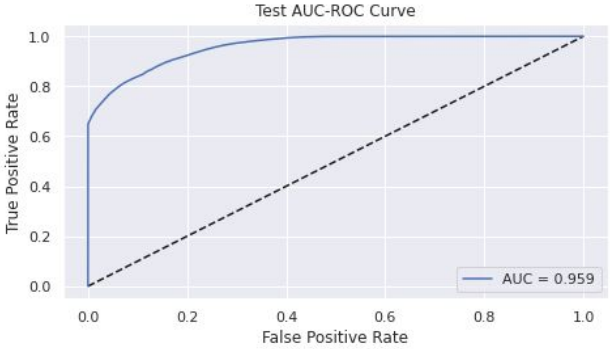



Train Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8687175786187346 |
| Precision | 0.8868576640961375 |
| Recall | 0.845227745830708 |
| F1 Score | 0.8655424264884545 |
| ROC-AUC Score | 0.8687141137101363 |
| Log Loss | 4.53437730113712 |



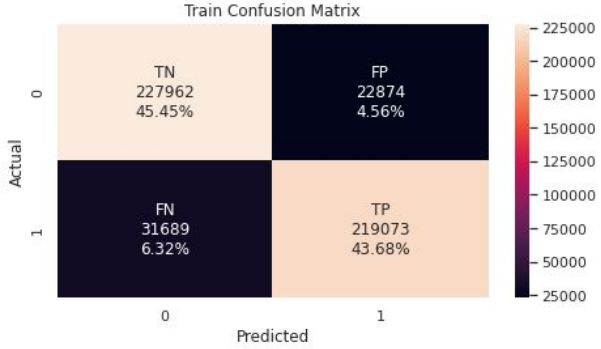



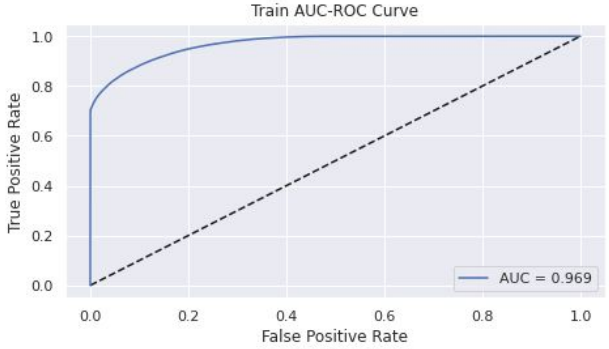

Test Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8685406698564593 |
| Precision | 0.885882014244408 |
| Recall | 0.846204431053242 |
| F1 Score | 0.8655887676728143 |
| ROC-AUC Score | 0.8685505598895567 |
| Log Loss | 4.540488008910933 |

- The best parameters found out to be : {'n_estimators': 200, 'min_data_in_leaf': 100, 'max_depths': 7.0, 'learning_rate': 0.01}

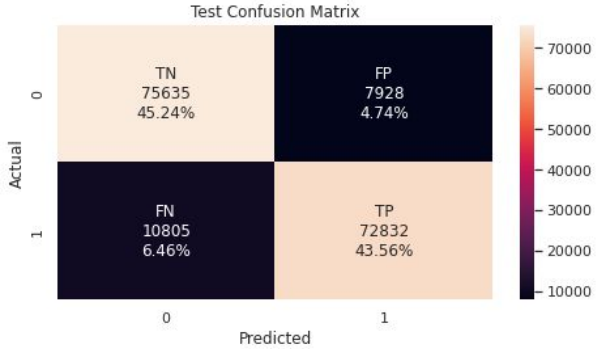
# CatBoost

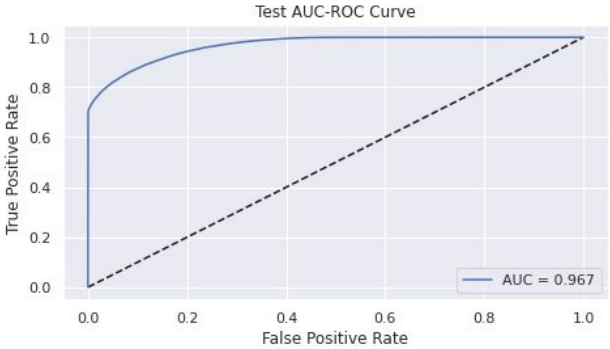



Train Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8912216555887383 |
| Precision | 0.9054586335023787 |
| Recall | 0.8736291782646494 |
| F1 Score | 0.8892591773237348 |
| ROC-AUC Score | 0.8912190605798044 |
| Log Loss | 3.757107377675211 |




Test Evaluation Metrics

| Metric | Score |
|---|---|
| Accuracy | 0.8879605263157895 |
| Precision | 0.9018325903912828 |
| Recall | 0.8708107655702619 |
| F1 Score | 0.8860502320601958 |
| ROC-AUC Score | 0.8879681198816928 |
| Log Loss | 3.869744242908424 |

- The best parameters found out to be : {'n_estimators': 150, 'max_depth': 7}

# MODEL EXPLAINABILITY

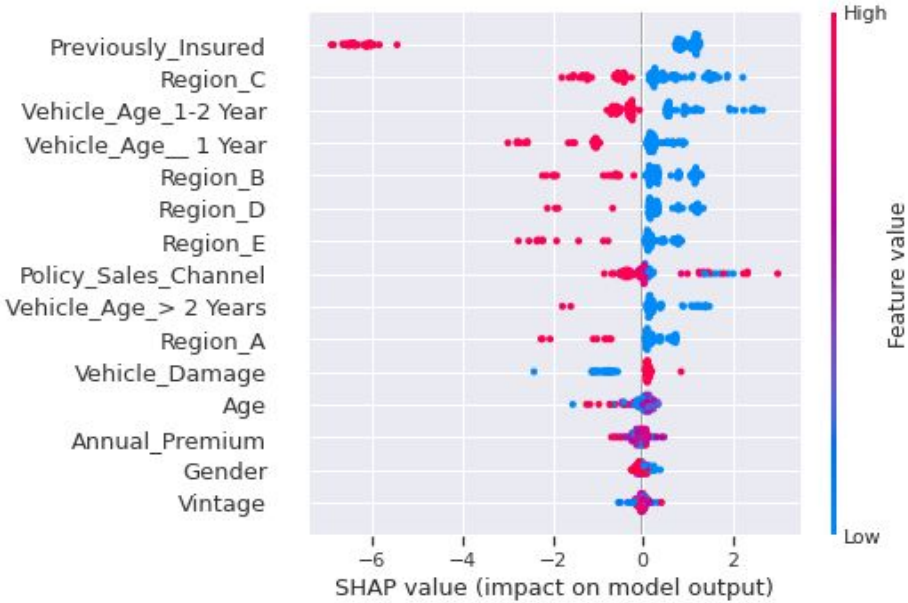
# Selecting Best Model and Model Explainability

- **XGBoost** can be chosen as **The Best Model** as it's overall evaluation scores are better compared to all the models with :
  - **Accuracy : 89.92%**
  - **F1 Score : 0.8981**
  - **ROC-AUC : 0.8992**
  - **Log Loss : 3.4803**
- **CatBoost** is not very far behind, so we can conclude it to be second best option.

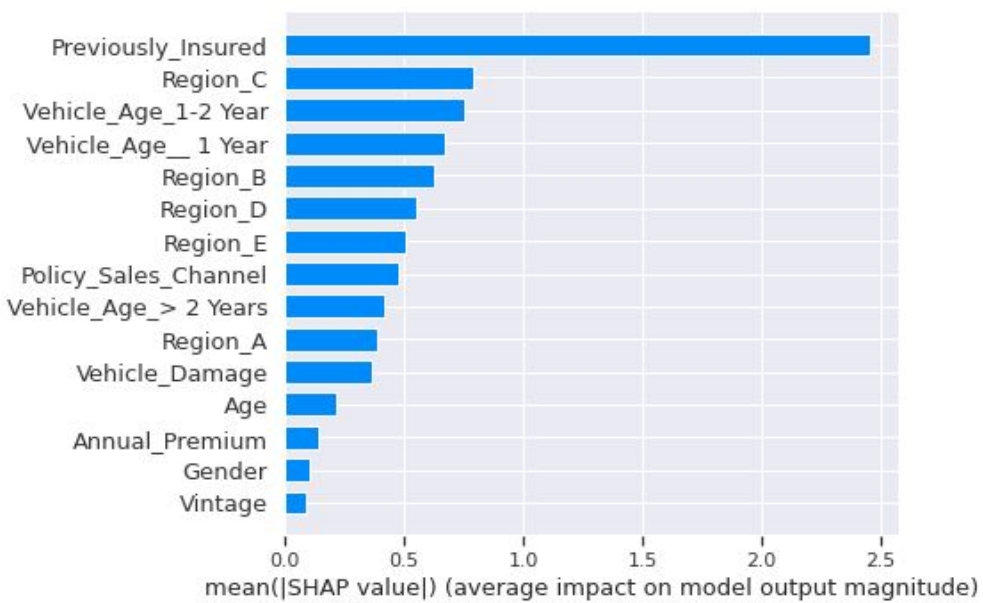
## SHAP Force Plot:

# Selecting Best Model and Model Explainability

## SHAP Summary Scatter Plot:




## SHAP Summary Bar Plot:



# Conclusion

- Initiating the task, we found that our raw data does not contain any missing or duplicate values. We then moved on to outlier removal and cleaned the dataset further.
- Deriving important business insights through EDA, we also performed feature engineering. This included Label encoding and One Hot encoding of categorical data, and removing redundant features.
- Our dataset was heavily imbalanced, so we applied SMOTE for handling class imbalance. We also scaled our data using MinMaxScaler for efficient model building.
- Further, moving on to ML model building step, we came up with evaluation metrics for different models and gauged their performances.
- Following Models were used:
    - Logistic Regression, Gaussian Naive Bayes, Decision Trees, Random Forest, XGBoost, LightGBM,CatBoost
- Out of which XGBoost gave us the best performance.
- We have successfully trained our model giving us an overall test performance of about 90%

# THANK YOU