# Open-IIT Data Analytics Report

## Team- PROGRESSIVE PREDICTORS

# CUSTOMER LIFE-TIME VALUE for Auto Insurance Company

# Contents



# INTRODUCTION

## 1.1 → Project Overview

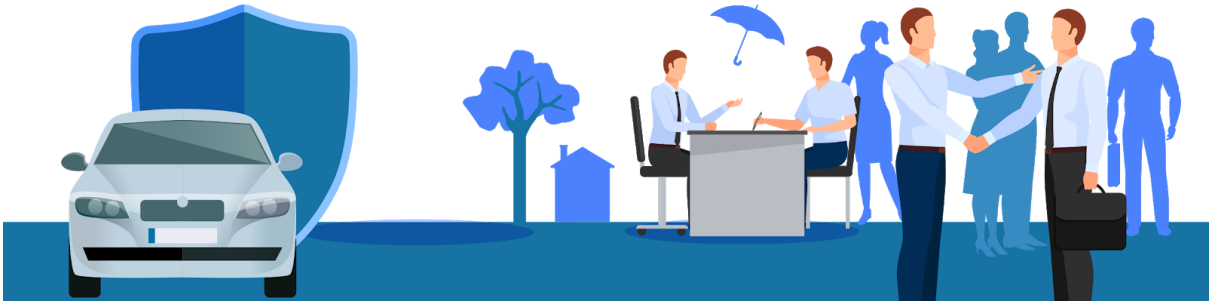
**Objective:**
To create a model to predict the customer lifetime value using a given set of features for an auto insurance company and also help the company to know the types of customers that would give it more revenue.

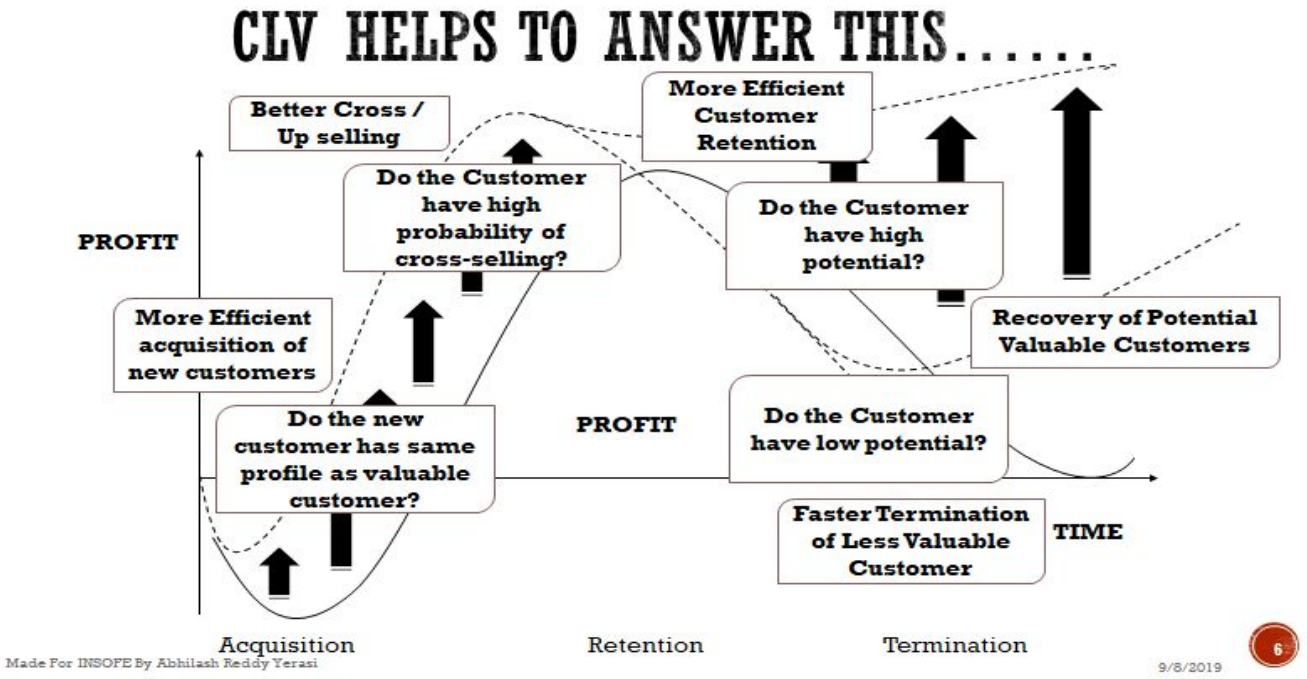
Data Set → Analyzing data → Creating an appropriate model after data preprocessing and feature Engineering → Predicting the CLV & targeting beneficial customers

| TERM | MEANING |
|---|---|
| AUTO INSURANCE | **Auto insurance is a contract between you and the insurance company that protects you against financial loss in the event of an accident or theft** |
| CUSTOMER LIFETIME VALUE | **It is a prediction of the net profit attributed to the entire future relationship with a customer.** |

## 1.2 → Problem Statement

For an Auto Insurance Company we predict the customer life-time value(CLV), CLV is the total revenue the client will derive from the entire relationship with a customer. Our aim is also to find the customer base which would be more beneficial.

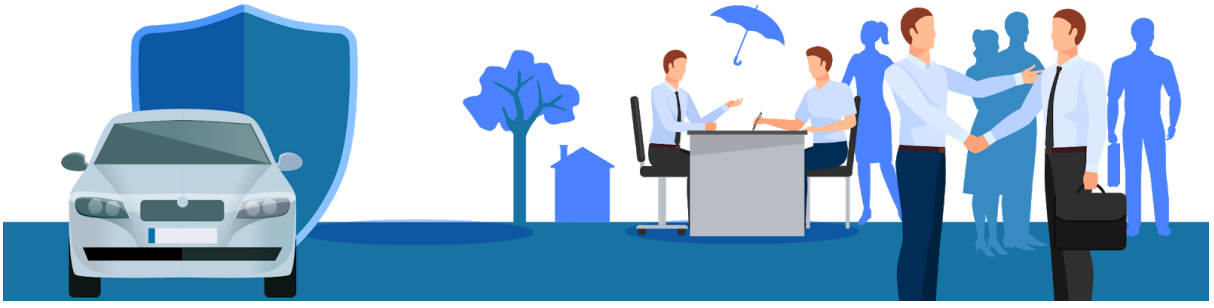





## 1.3 → Data Description

**There are 24 columns in the given data set.**
**We have described the important ones below :**

| COLUMN NAME | DESCRIPTION |
|---|---|
| Customer Lifetime value | The total revenue the client will derive the entire relationship with a customer. |
| Coverage | The amount of risk or liability that is covered for an individual or entity by way of insurance services. |
| Location Code | The region of the customer i.e. urban, suburban or rural. |
| Months Since Last Claim | The number of months it has been since a customer last claimed their insurance. |



| | |
|---|---|
| **Months Since Policy Inception** | **Number of months since the policy was taken by the customer.** |
| **Number of Open Complaints** | **The number of complaints filed by the customers for claiming insurance.** |
| **Policy Type** | **The type of policies taken i.e. corporate, personal, or special.** |
| **Sales Channel** | **A method of distribution used by a business to sell its products, that is agent,call center, etc.** |
| **Vehicle Class** | **Class of the vehicle i.e. two-door, four-door, SUV, luxury SUV, Sports or Luxury** |

## 1.4->EVALUATION METRICS:

**RMSE:**

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how to spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

**FORMULA:**

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (P_i - O_i)^2}{n}}$$

## MAPE:

It is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning. It usually expresses accuracy as a percentage, and is defined by the formula:

relative or percentage error

First we need to define a relative or percentage error as

$$\mathrm{PE}_t = \left(\frac{Y_t - F_t}{Y_t}\right) \times 100. \tag{2.16}$$

Then the following two relative measures are frequently used:

mean percentage error

$$\mathrm{MPE} = \frac{1}{n}\sum_{t=1}^{n} \mathrm{PE}_t \tag{2.17}$$

mean absolute percentage error

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n} |\mathrm{PE}_t| \tag{2.18}$$

# ANALYSIS

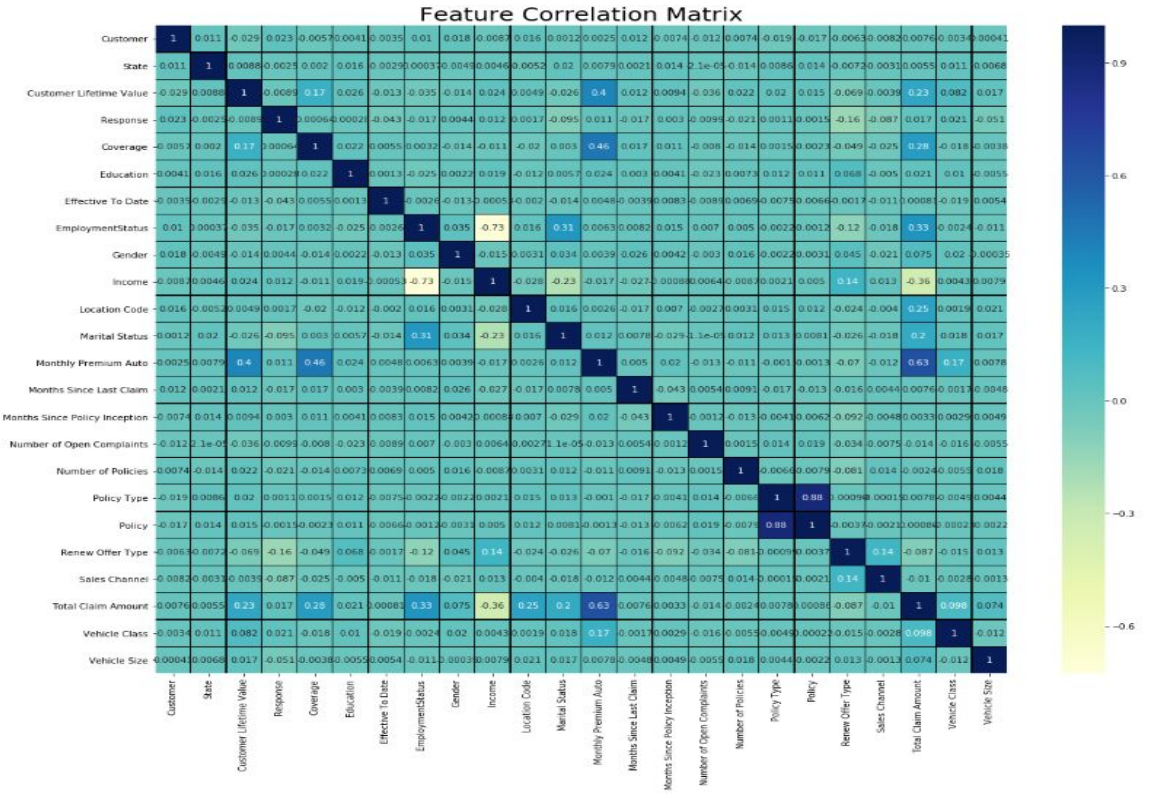
---

## 2.1→ Data Explorations and Analysis:

This is our first step in data analysis and here we would typically find the important characteristics and patterns in our data set which would help us to draw some hypotheses about what our data is likely to support.

### CORRELATION MATRIX:

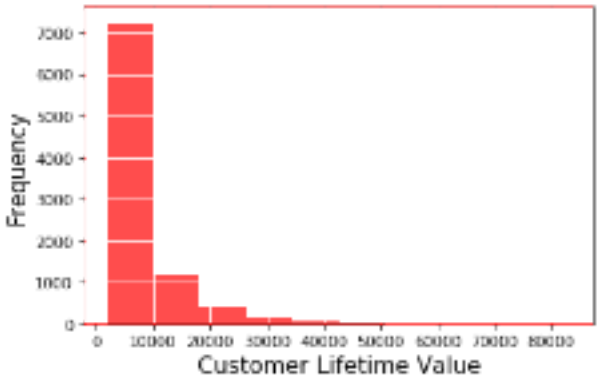
It is a table showing the correlation between different variables. Each cell in the table shows correlation between two variables.



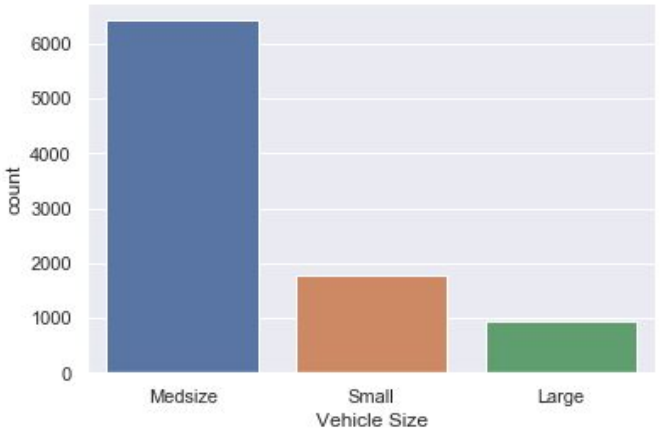

It gives us a direction towards data preprocessing and exploratory analysis.

# Graphs for Data Exploration

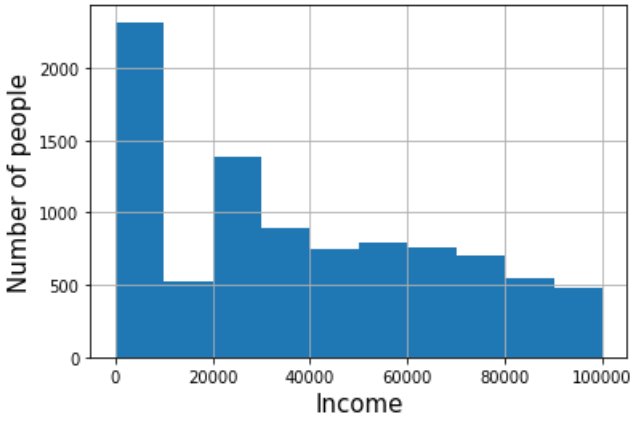
---




It is observed that the CLV is skewed
Seeing this we can apply normalization, the sigmoid, Log and try to achieve the best output.




It is observed that the Midsize the vehicle is most common. So we know that the customers with midsize vehicles are more inclined to get insurance.

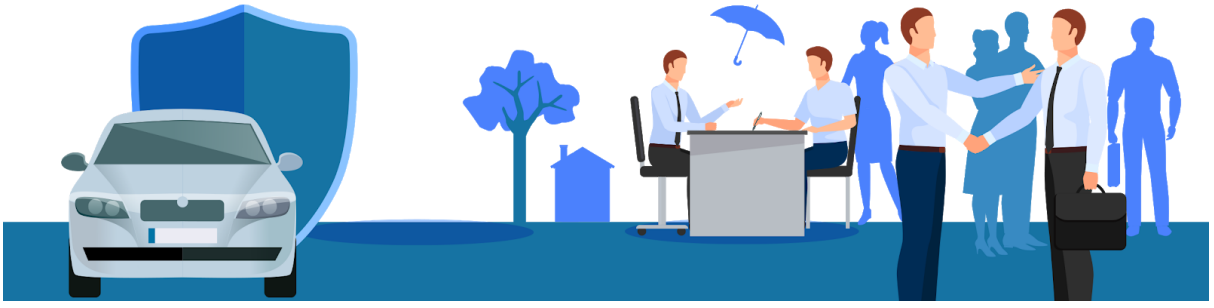



There is a number of unemployed people in the given dataset. So we can infer that students are more inclined to get insurance.

## 2.2→Feature Engineering

---

We apply feature engineering to prepare a proper input dataset, compatible with machine learning algorithm so as to improve the performance of machine learning models



| | |
|---|---|
| Label Encoding | It encodes labels with a value between 0 and n classes where n is the number of distinct labels. |
| One Hot Encoding | This is one of the most common encodings.It spreads the values in a column to multiple flag columns and assigns 0 or 1 to them. |
| Binning | It is done to make the model more robust and prevent overfitting<br>We have dropped the columns- |
| Log Transform | As our data is skewed we apply log transform to make the distribution normal. It also decreases the effect of outliers. |
| Scaling | We did standardization and normalization on our data as the values of CLV have a very high range as compared to the other columns |

# Algorithms & Techniques

## 3.1→MODEL SELECTION & PERFORMANCE:

| Models | RMSE | MAPE |
|---|---|---|
| **Linear Regression** | **6206.445852457786 RMSE** | **58.92278512311637 MAPE** |
| **Decision Tree Regression** | **4900.9751404218105** | **11.960203207519376** |
| **XGBoost** | **4073.562562223434** | **11.48024232434435** |
| **Random Forest (no. of trees = 1000)** | **3741.852642288239** | **11.014127435360363** |
| **Random Forest (no. of trees = 2000)** | **3746.4765147357334** | **11.007724840486741** |

## RANDOM FOREST REGRESSOR

**We now attach the snippet of the model which gave us the minimum RMSE:**

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 2000, random_state = 0)
rf.fit(x_train, y_train)
```

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. If you want to read more on Random Forests, I have included some reference links which provide in-depth explanations on this topic.

Here, we have chosen two hyperparameters; random_state and n_estimators, to be optimized. According to sklearn documentation, n_estimators, the number of trees in the forest. Ideally, you can expect a better performance from your model when there are more trees.
However, you must be cautious of the value ranges you specify and experiment using different values to see how your model performs.

```
y_pred = rf.predict(x_test)

from sklearn.metrics import mean_squared_error
from math import sqrt
rms = sqrt(mean_squared_error(y_test, y_pred))
```

## 3.2→ PRECAUTIONS:

We prevent **OVERFITTING** by selecting appropriate number of leaves in our random forest model.

We could have a better prediction of the validation data set by increasing the number of leaves but in that case, the model may fit the training set well but it would not make good predictions on the test data or new data.

We also prevent **UNDERFITTING** by not making a very simple model that would not give a good output. Simple model here refers to a model where we drop many of the columns and work with only a few.

## 3.3→ UNCERTAINTY

The model that we propose a purely mathematical one where we don’t take into consideration the human factors and hence our prediction is only a best guess taking into consideration our dataset.

We have various situations that we cannot account for like political changes,new insurance companies or laws for/or against insurance companies.

---

# 5.→ CONCLUSION

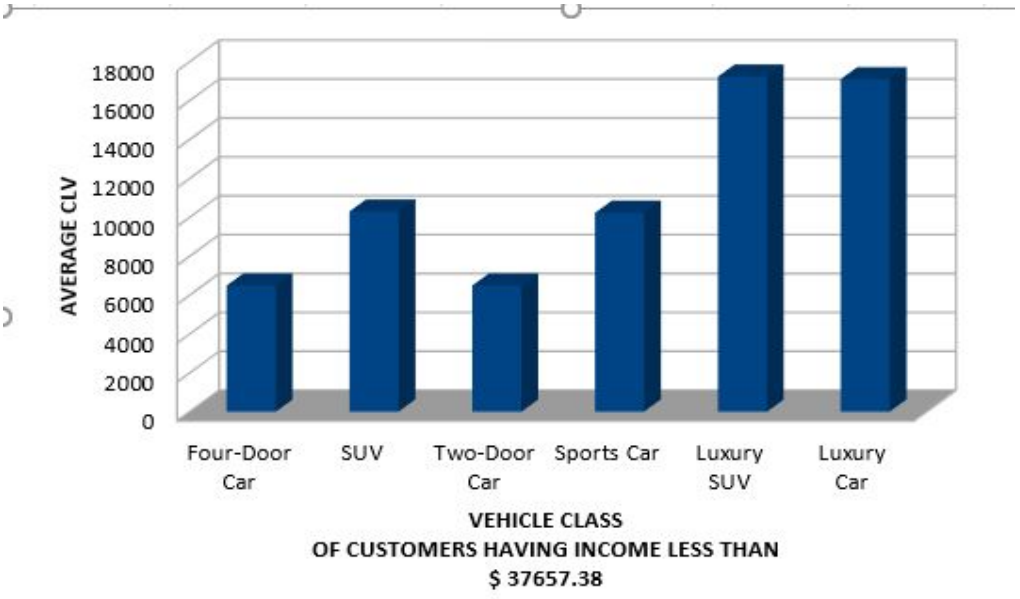
## Business Recommendations:

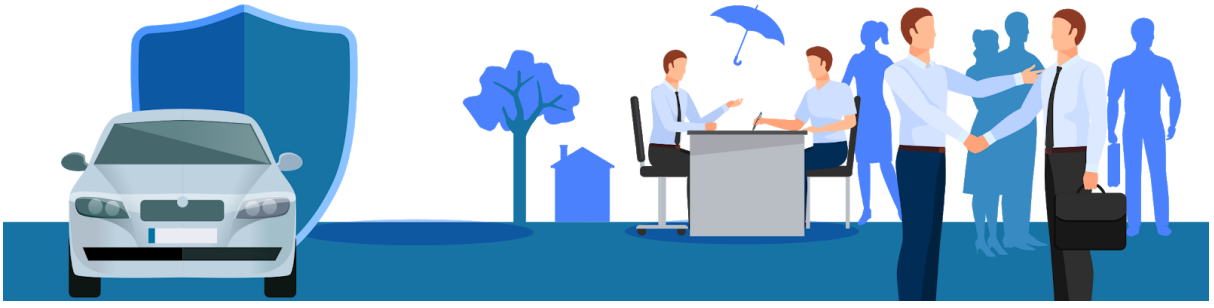
According to the various models applied with their variations, which we have included in our feature engineering, we have come to the conclusion that targeting people who satisfy the following two conditions:

1. Income lower than the mean,
2. And has vehicle size is medium or large

These categories of people comparatively require auto insurance and hence the company should target these customers.

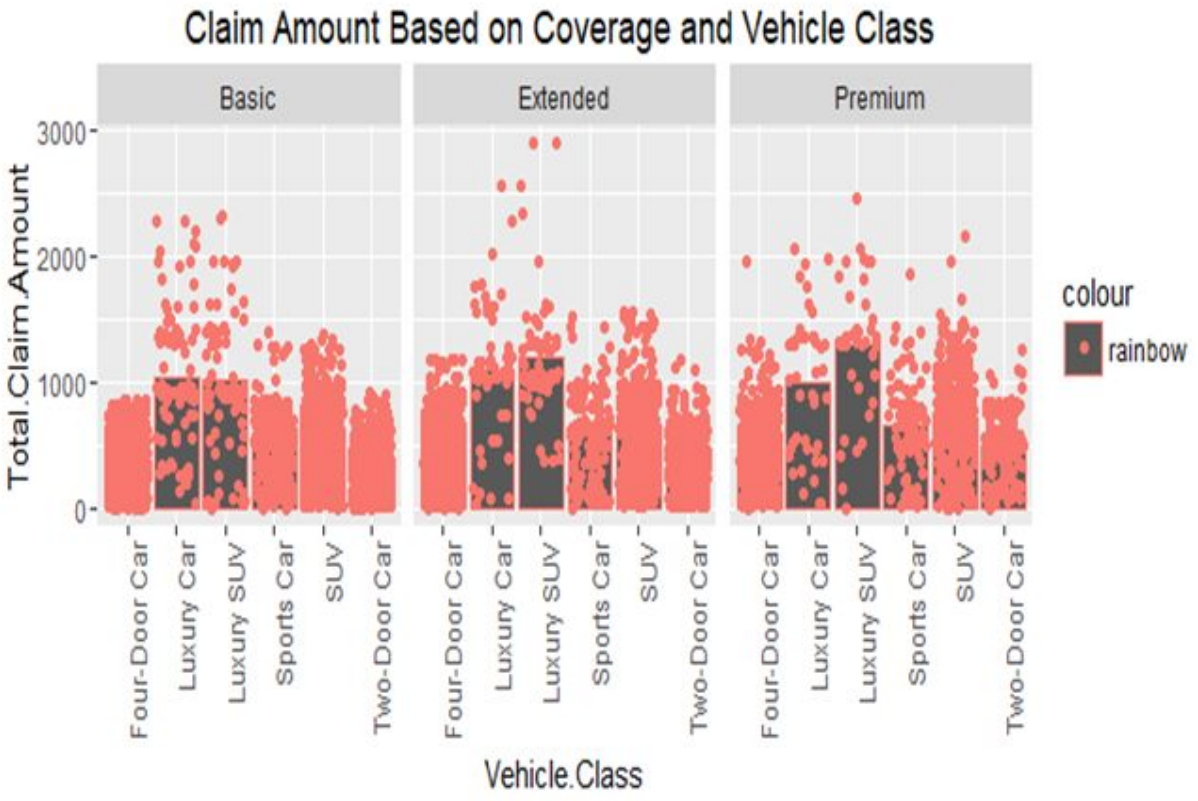
Applying the above-mentioned models, we find that the best RMSE value is given by Random Forest Regressor and hence we use the same, with tweaking the n_estimators hyperparameter.

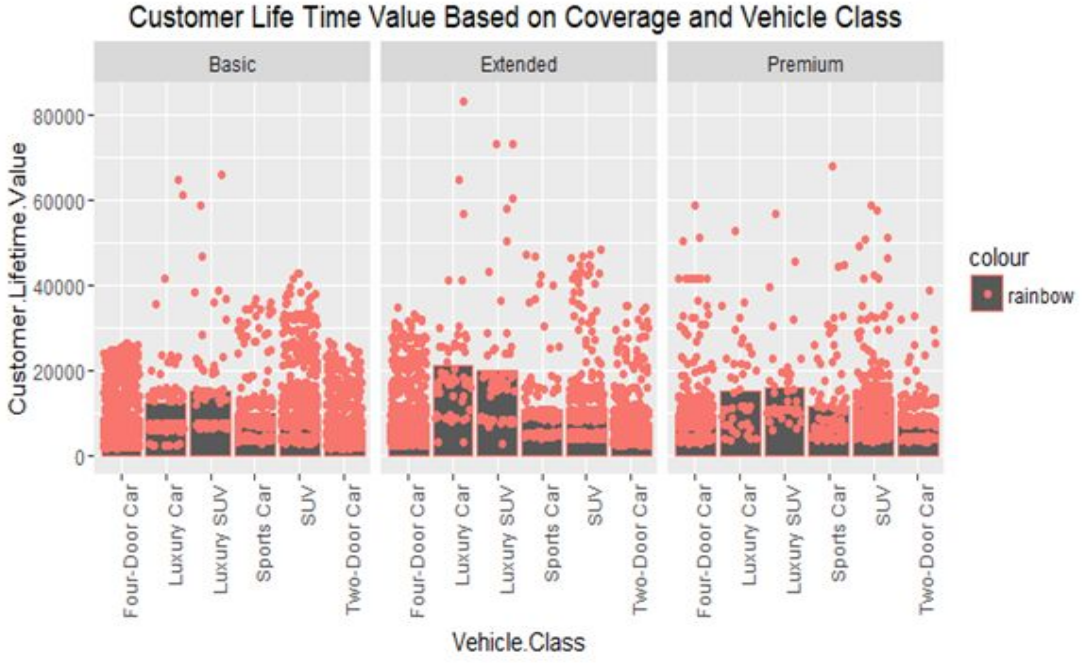


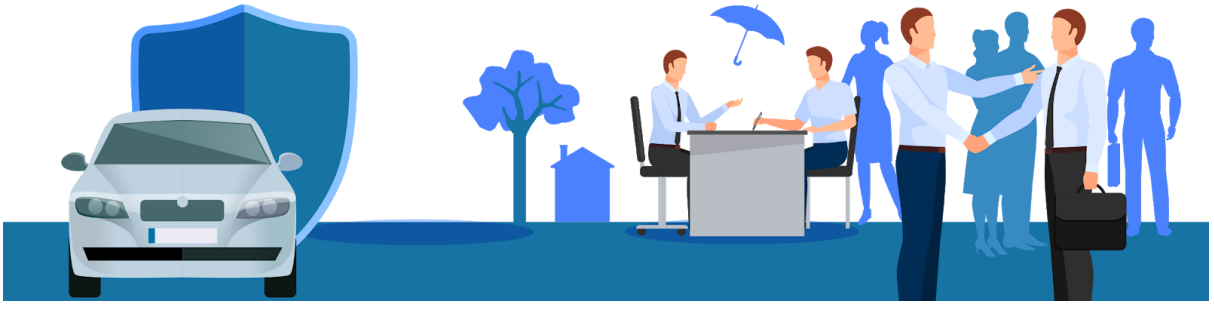



**Insights based on segmenting customer groups:**

- The Customer who is using Four-Door Car and Two Door cars is more inclined to coverages and even the Loss incurred by them is more compared to others.
- After Clustering we can easily segregate the customers into 4 groups and they were clearly different from other groups.
- Group 4 Customers are the most profitable customer in comparison with the customer lifetime value but the no. of customer count is less.
- Even Group 3 Can be Bought into Group 4 by giving offers to them and even new attracting policies.
- Concluding that the random Forest model is stable in order to predict the CLTV.



Claim Amount Based on Coverage and Vehicle Class
Basic
Extended
Premium
Total.Claim.Amount
3000
2000
1000
0
Four-Door Car
Luxury Car
Luxury SUV
Sports Car
SUV
Two-Door Car
Vehicle.Class
colour
rainbow


Customer Life Time Value Based on Coverage and Vehicle Class
Basic
Extended
Premium
Customer.Lifetime.Value
80000
60000
40000
20000
0
Four-Door Car
Luxury Car
Luxury SUV
Sports Car
SUV
Two-Door Car
Vehicle.Class
colour
rainbow



# 6→ Annexure

## Software/Libraries Stack:

1. **Python 3.6** - Language of choice
2. **Pandas** - for Handling files
3. **Numpy** - for complex numerical analysis
4. **Seaborn** - plotting advance visualizations
5. **Matplotlib** - plotting visualizations
6. **Sklearn** - for making machine learning models
7. **XGBoost** - for using boosting models
8. **Jupyter** Notebook

## References:


1. **Label Encoder** :
   https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
2. **One Hot Encoder** :
   https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
3. **Heatmap** :
   https://seaborn.pydata.org/generated/seaborn.heatmap.html
4. **Random Forest Regressor** :
   https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb
5. **Decision Tree** :
   https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
6. **xgboost Regressor** :
   https://xgboost.readthedocs.io/en/latest/python/python_api.html