**Case Study – Report**

**Background**

***Objective****: Predict Claim Frequency for the given portfolio based on the historical Claim Count, Exposure, and Features.*

The data was checked for validity, completeness, consistency, and uniformity. The Summary Statistics are shown below.

**Table

Description automatically generated**

Figure 1: Summary Statistics

**Exploratory Analysis**

The relationships between the target variable and the features have been visualized below.

*Chart

Description automatically generated*

Figure 2: Relationships between Features and Target

The following relationships can be hypothesized:

1. Average Claim Frequency is greater for Females than Males.
2. Claims Frequency increases with weight of the vehicle.
3. Claims Frequency decreases with age of the driver.
4. Claims Frequency increases with age of the car.

However, the correlation matrix between the features and the Claim Frequency shows very small correlations. Therefore, even if Null Hypotheses are rejected and statistically significant relationships are found, the effect size is expected to be very small.

**Methodology**

Quadratic and Interaction Features were Engineered. Two strategies for feature selection were considered – 1. Eliminating features which showed high correlation with other features and 2. Retaining features with high importance scores in a preliminary tree-based model. However, since we intended to use Tree Methods, it was decided to retain all the features because:

1. Decision Trees are by nature immune to multicollinearity and will select the important features.
2. We are interested in predictions, not in causality.
3. Our model contains relatively small number of features anyway (ie. narrow dataset). So, reducing processing time is not a priority.

Two methods are commonly used to model Claim Frequency:

1. With the response as the sum of claim counts (i.e. Counts), and passing exposure as *offset*.
2. With response as claims per exposure (i.e. Claim Frequency), and passing exposure as *sample\_weight*.

The first method is not possible with Decision Trees, hence we opt for method 2.

**Modelling**

The process followed was:

1. For each algorithm, build model and find best parameters with 10-fold Cross Validation in a single step using scikit=learn’s GridSearchCV for the Training data (80%).
2. Test the selected best version of the model on the Test data and score it using Mean Square Error (MSE) and the Root Mean Square Error (RMSE, more interpretable than MSE).
3. Compare the MSE/RMSE of the different selected models to find the best model.

To find the optimal size of the tree which does not overfit or underfit on the data (bias-variance trade-off), GridSearchCV was used for Hyperparameter Tuning.

3 models (Decision Tree and two Gradient Boosted Models) were trained and tested. The MSE and RMSE of the tested models are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Decision Tree** | **Sklearn’s Gradient Boosted Regressor** | **XGBoost Regressor** |
| **MSE** | 0.073 | 0.073 | 0.073 |
| **RMSE** | 0.270 | 0.269 | 0.269 |

Figure 3: Errors for the Candidate Models

The features which showed importance scores greater than the mean importance score in the xgboost model are summarized below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **gender** | **weight** | **age** | **carage** | **Distance\*weight** | **Distance\*carage** | **Weight\*carage** |
| 0.096303 | 0.109485 | 0.157438 | 0.0745 | 0.086788 | 0.08831 | 0.211726 |

Figure 4: Features showing Importance greater than the mean importance score in the xgboost model

However, since we have left in many correlated features, the importance ratings could have been split and not be perfectly representative of the true importance of the features. Furthermore, tree models tend to overestimate the feature importance of categorical features (gender in our dataset).

**Conclusion**

There was very little difference between the performance of the models with the Gradient Boosted Models performing slightly better (RMSE = 0.269).