

CSCE 5380: Data Mining Project Final Report

Categorizing Car Insurance Claims: A Predictive Analysis Study

Shashidhar Kalapatapu

Sai Teja Shaga

shashidharkalapatapu@my.unt.edu

saitejashaga@my.unt.edu

Gowtham Sai Lendhra Narayanam

11545863

11613473

gowthamsailendhranarayanam@my.unt.edu

11709752

Aditya Vadrevu

Meghana Vagdevi Korada

adityavadrevu@my.unt.edu

meghanavagdevikorada@my.unt.edu

11601517

11604477

Abstract:

Car insurance claims are one of the most frequently encountered types of insurance claims. From an insurance company's perspective, handing out insurances to customers can be a risk if the customer is likely to claim the insurance within a year of his/her policy tenure. For such reasons, Insurance Companies need a system to identify such potentially risky customers, and use their past insurance history to predict the possible claims if such customers were given insurance. This project develops a predictive model for car insurance by analyzing a comprehensive dataset of policyholder information. We addressed the growing complexity of insurance risk assessment by building a model to forecast the likelihood of policyholders filing claims within the next six months.

We framed the problem as a classification task, using machine learning techniques to classify policyholders into two categories: those who will file a claim and those who will not in the next six months. To ensure the accuracy of our results, we have utilized several evaluation metrics including accuracy, precision, recall, and F1-score. Additionally, we implemented cross-validation techniques to mitigate overfitting.

The project utilized the "Car Insurance Claim" dataset obtained from Kaggle repositories. Since limited research existed on this specific dataset, we leveraged similar models built on other datasets to benchmark the general performance of our model. A core focus of the project was maintaining a clean dataset. We used data cleaning techniques to ensure data quality and reduce the number of features to the most relevant ones for model building. We implemented the project in Python, using libraries such as Pandas

for data manipulation, NumPy for numerical computations, Scikit-learn for machine learning algorithms and model development, and Seaborn for data visualization.

The project explored various classification algorithms including Decision Trees, Logistic Regression, Random Forest, Light Gradient Boosting, Naive Bayes, K Nearest Neighbors, Multi-layer Perceptron neural networks, Decision Tree, Random Forest, AdaBoost, XGBoost, Gradient Boosting, LightGBM, and CatBoost. We used GridSearchCV and Optuna for hyperparameter tuning of Logistic Regression, MLP Classifier, and LightGBM, but observed minimal improvements in accuracy. This project successfully discovered machine learning models that can assist insurance companies in better risk assessment and management. By improving risk assessment, insurance companies can enhance customer service and achieve greater profitability. After much experimentation and evaluation, we achieved **an accuracy of 93.39%** using the Light Gradient Boosting algorithm, Logistic Regression, and Multi Layer perceptron .

1. Introduction:

The accurate prediction of car insurance claims is critical for insurance companies in effectively managing their resources and claim procedures. Traditional methods of risk assessment can struggle to keep pace with the growing complexity of factors influencing insurance claims. This project addresses this challenge by developing a machine learning model to predict the likelihood of policyholders filing claims within the next 6 months.

This project is expected to deliver insights on which machine-learning model can assist insurance companies in improving their risk assessment and management processes. By enhancing risk assessment accuracy, insurance companies can achieve better customer service and greater profitability. Additionally, the project aims to contribute to the field of car insurance claim prediction by exploring a specific dataset and comparing the performance of various classification algorithms.

Although not much related work was not found, similar work was done, such as to predict the cost of damages from car crash images, or proposing insurance policies to new customers using their past insurance data.

2. Experiment Methodology:

2.1 Dataset:

The dataset chosen for this project is the “**Car Insurance Claim Prediction**” Dataset from Kaggle [1]. This dataset was originally taken from a Dataverse Hackathon Competition hosted by Analytics Vidhya in 2022 [2]. The dataset contains 58592 rows and 44 columns. This includes columns like age_of_car, age_of_policyholder, make of the car, engine, power, displacement, max_torque, speed_alert, ncap_rating. The target attribute ‘**is_claim**’ which is a binary outcome (0 - for No claim, 1 - for Yes) indicates whether the policyholder will file a claim in the next 6 months or not.

Complete List of Attributes :

Attribute	Description
policy_id	Policyholder ID
policy_tenure	Policy time period/duration
age_of_car	Normalized age of the car in years
age_of_policyholder	Normalized age of policyholder in years
area_cluster	Policyholder's area cluster
population_density	Population density of the city, where the policyholder is residing
make	Company which designed the car
segment	Car's Segment (A/ B1/ B2/ C1/ C2)
model	Car encoded name
fuel_type	Type of fuel required for the car
max_torque	Maximum Torque produced by the car in Nm@rpm
max_power	Maximum Power produced by the car bhp@rpm
engine_type	Car Engine type used
airbags	No. of airbags installed in the car
is_esc	Electronic Stability Control (ESC) is present in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_adjustable_steering	Steering wheel of the car is adjustable or not. (Boolean Flag 1/0 -> Yes/No)
is_tpms	Tyre Pressure Monitoring System (TPMS) is present in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_parking_sensors	Parking sensors are present in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_parking_camera	Parking camera is present in the car or not. (Boolean Flag 1/0 -> Yes/No)
rear_brakes_type	Type of brakes installed in the rear side of the car
displacement	Total Engine displacement of the car in cc
cylinder	No. of cylinders in the engine of the car

transmission_type	Car's Transmission type
gear_box	Total No. of gears in the car
steering_type	The type of power steering present in the car.
turning_radius	The space in meters, the car needs to make a turn (Meters)
length	Car Length in Millimetre
width	Car Width in Millimetre
height	Car Height in Millimetre
gross_weight	Max weight in Kgs allowed in fully loaded car including the passengers, cargo, and equipment.
is_front_fog_lights	Front fog lights are available in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_rear_window_wiper	Rear window wiper is available in the car or not.(Boolean Flag 1/0 -> Yes/No)
is_rear_window_washer	Rear window washer is available in the car or not.(Boolean Flag 1/0 -> Yes/No)
is_rear_window_defogger	Rear window defogger is available in the car or not.(Boolean Flag 1/0 -> Yes/No)
is_brake_assist	Brake assistance feature is available in the car or not.(Boolean Flag 1/0 -> Yes/No)
is_power_door_lock	Power door lock feature is available in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_central_locking	Central locking feature is available in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_power_steering	Power steering feature is available in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_driver_seat_height_adjustable	Height adjustment of the driver seat is available in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_day_night_rear_view_mirror	Day & night rearview mirror is present in the car or not. (Boolean Flag 1/0 -> Yes/No)
is_ecw	Engine Check Warning (ECW) feature is available in the car or not. (Boolean Flag 1/0 -> Yes/No)

is_speed_alert	Speed alert system is available in the car or not. (Boolean Flag 1/0 -> Yes/No)
ncap_rating	The Safety rating given by NCAP program (rated out of 5)
is_claim	Target Variable: Indicates whether the policyholder will file a claim in the next 6 months or not. (Boolean Flag 1/0 -> Yes/No)

2.2 Software and Tools

2.2.1 Language and Libraries

The implementation of this data mining project is carried out by using the **Python programming language**. Python3 offers an extensive collection of libraries and is popularly used for Machine Learning, Artificial Intelligence, and Data Mining [3]. Often these fields blend in together for certain problem statements such as ours which is Car Insurance Claim Prediction.

The Python3 packages used for this project:

- **NumPy (1.25.2):** For handling and manipulating high performance multi-dimensional arrays. Used for Data preprocessing, data manipulations, data holding.
- **Pandas (2.0.3):** For accessing, modifying, and analyzing DataFrame Objects which are built using Series Objects from the NumPy Library.
- **Matplotlib (3.7.1):** For visualizing data with plots like bar plot, pie chart, scatter plot etc.
- **Seaborn (0.13.1):** Another visualizing library which offers much richer and detailed visualizations, built on top of matplotlib library.
- **Scikit-learn (1.2.2):** All-in-one library for data preprocessing, data manipulation, data encoding, designing predictive model with different algorithms, model evaluation, hyperparameter tuning
- **XGboost (2.0.3):** Optimized implementation of XGBoost Algorithm.
- **Light GBM (4.1.0):** Boosting framework with tree based learning algorithms.
- **Catboost (1.2.5):** High performance gradient boosting library on Decision Trees.
- **Optuna (3.6.1):** Another library for Hyperparameter tuning which offers better grid search capabilities for Light GBM Algorithm.

2.2.2 Execution Environment and Setup:

For running and evaluating the project, **Jupyter Notebook (6.4.5)** was used. Jupyter Notebook is a web based Python Computing environment, where blocks of code can be executed in mini shell based cells [4]. Installing Jupyter is pretty easy, it is included when installing Python from the Anaconda distribution.

While the Jupyter Environment was mostly used to run the code locally, we have also used **Google Colab**. Colab offers a similar environment as Jupyter but there's absolutely no installation needed [5].

Except for connecting to Google Drive Storage and installing certain packages, Colab makes it easy to run and monitor the model's performance.

2.3 Implementation Methodology

The implementation process of our project can be described into 6 phases as shown in the implementation cycle from Figure 1.1. The process is not completely linear, as with any machine learning problem, we often juggled with Data preprocessing, Data Analysis and Predictive Modeling phases.

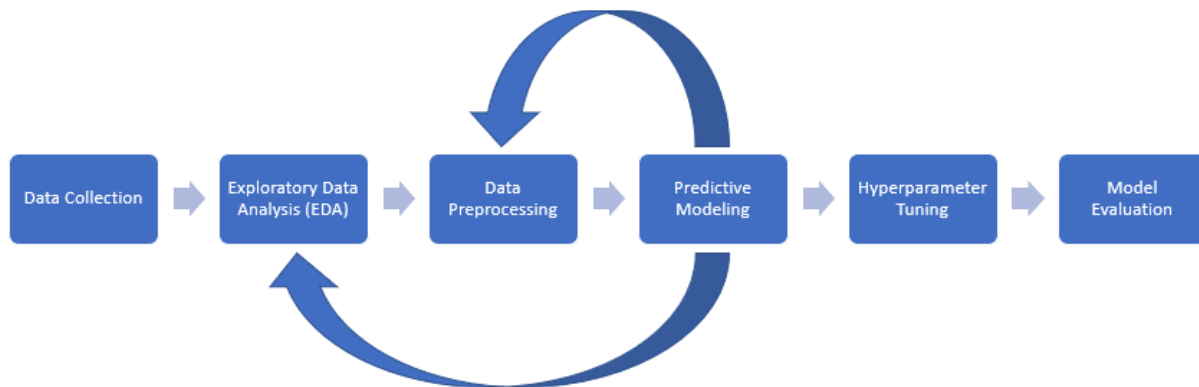


Figure 1.1: Project Implementation Cycle

2.3.1 Data Collection:

The dataset as described previously is from the Kaggle repository. The dataset contains three files, train.csv, test.csv, and sample_submissions.csv. Although, a test.csv was provided in the kaggle webpage. This test.csv was a part of the competition, and the actual values for the target variable in the test.csv were not provided. Thus, the train.csv was used for training and testing. The split of the training and testing dataset was 75-25%. This split was achieved using the train_test_split() method from scikit-learn library.

```

# Importing the train csv file

# the folder where the dataset and the project files reside
# path = "C:/Users/admin/CSCE 5380 Data Mining Project/data/"
path = "/content/drive/MyDrive/car_insurance_data/"

# lets load the train dataset into a dataframe called 'df'.
df = pd.read_csv(path + "train.csv")

# the first 5 rows of the dataframe
df.head()

```

	policy_id	policy_tenure	age_of_car	age_of_policyholder	area_cluster	population_density	make	segment	model	fuel_type	...	is_brake
0	ID00001	0.515874	0.05	0.644231	C1	4990	1	A	M1	CNG	...	
1	ID00002	0.672619	0.02	0.375000	C2	27003	1	A	M1	CNG	...	
2	ID00003	0.841110	0.02	0.384615	C3	4076	1	A	M1	CNG	...	
3	ID00004	0.900277	0.11	0.432692	C4	21622	1	C1	M2	Petrol	...	
4	ID00005	0.596403	0.11	0.634615	C5	34738	2	A	M3	Petrol	...	

5 rows x 44 columns

Figure 1.2: Data Loading

Once the data is collected and stored as DataFrame Objects (Figure 1.2), we moved to the Exploratory Data Analysis Phase.

2.3.2 Exploratory Data Analysis:

Exploratory data analysis usually involves identifying data outliers, missing values, hidden patterns, and evaluating the distributions of attributes.

Out of the 44 columns, there were 28 columns of object data type, 12 columns were int64 data type, and 4 float64. We found this information using the df.info() method (Figure 1.3).

```

# Column Information
df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58592 entries, 0 to 58591
Data columns (total 44 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   policy_id                             58592 non-null  object
1   policy_tenure                         58592 non-null  float64
2   age_of_car                            58592 non-null  float64
3   age_of_policyholder                   58592 non-null  float64
4   area_cluster                          58592 non-null  object
5   population_density                    58592 non-null  int64
6   make                                  58592 non-null  int64
7   segment                               58592 non-null  object
8   model                                 58592 non-null  object
9   fuel_type                             58592 non-null  object
10  max_torque                            58592 non-null  object
11  max_power                             58592 non-null  object
12  engine_type                           58592 non-null  object
13  airbags                               58592 non-null  int64
14  is_esc                                58592 non-null  object
15  is adjustable steering                 58592 non-null  object

```

Figure 1.3 Column Information

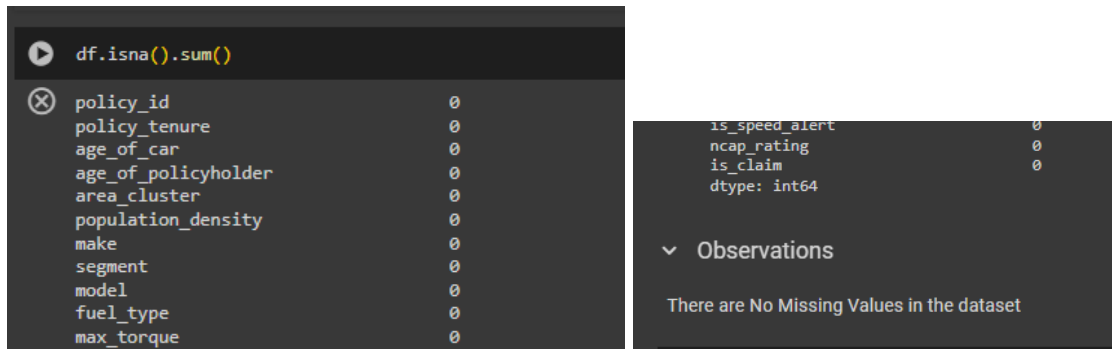


Figure 1.4: No Missing values

After concluding that there were no missing values present in the dataset (Figure 1.4), we moved to take a closer look at the statistical distribution of the columns (Figure 1.5). We found that for the ‘population_density’ column, there's a noticeably large difference between the 75% percentile and the max value. It could indicate the possibility of outliers.

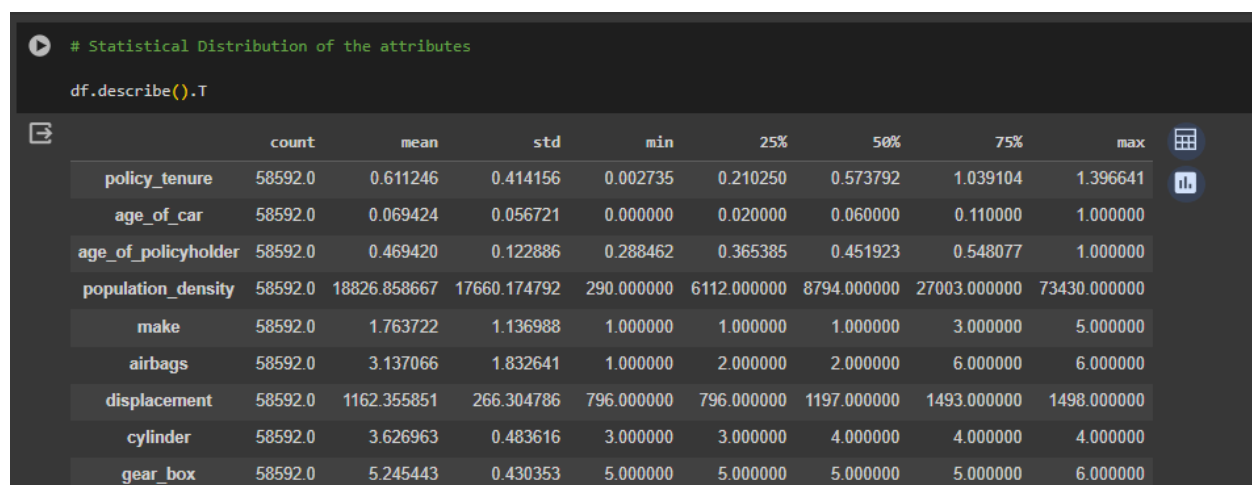


Figure 1.5: Statistical Distribution of the Attributes

We also found out that for the target variable, there was an imbalance with the class labels distribution. Out of 58592 samples, only 6.4% had a class label ‘1’ (yes) for ‘is_claim’ attribute. I.e. only 6.4% of the policyholders claimed the insurance within 6 months (Figure 1.6).

While Exploring the unique values for the columns, policy_holder and policy_tenure had all values as unique. If all the values are unique, then the information gained from these attributes are very less, so they are simply dropped and not used for predictions.

The columns were then split into categorical and numerical attributes to get a detailed look of their distributions.

Distribution of 'is_claim' Target Variable

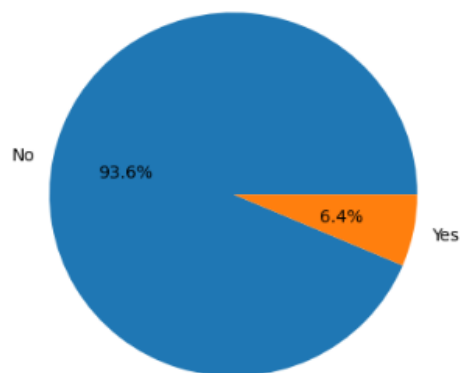


Figure 1.6: Distribution of is_claim variable

2.3.2.1 Numerical Attribute Analysis:

For the three numerical attributes [age_of_car, age_of_policyholder, population_density], a histogram plot was generated using the seaborn library (Figure 1.7).

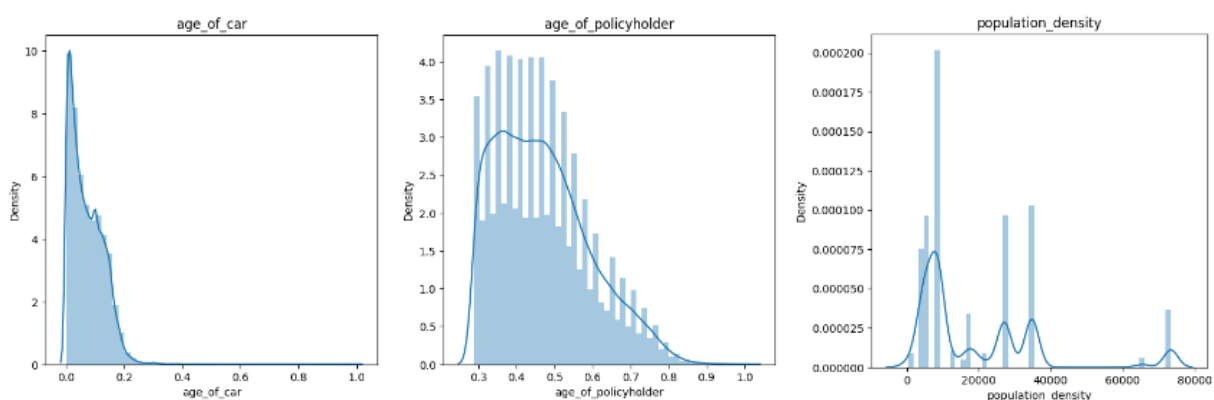


Figure 1.7: Histogram Plot for Numerical Attributes

We can see that for the 'age_of_car' column, the data is heavily skewed to the right. This implies that the majority of the cars are quite new and their age is less than 2-3 years. While for the 'Age_of_policyholder', the data is skewed to the right as well but only slightly. Possibly indicating that the most of the policyholders are in their teenage to mid 30 years. However, The population densities have different peaks, highest peaks indicating the most populated cities where the policyholders are from.

Plotting a Box plot for the same three columns as seen in Figure 1.8, reveals that there are plenty of outliers in them. But as the data is heavily skewed for age columns, most of them were not dropped. For the age_of_car, the threshold was set to 0.6 and the data is dropped if the age was greater than 0.6, then we re-normalize the data with 0.6 as the maximum. For the age_of_policyholder, the outliers are not that

distant from the max. So, we did not drop them. For the `population_density`, the data as seen before is not normalized. So no need to remove the outliers here, we will simply normalize it.

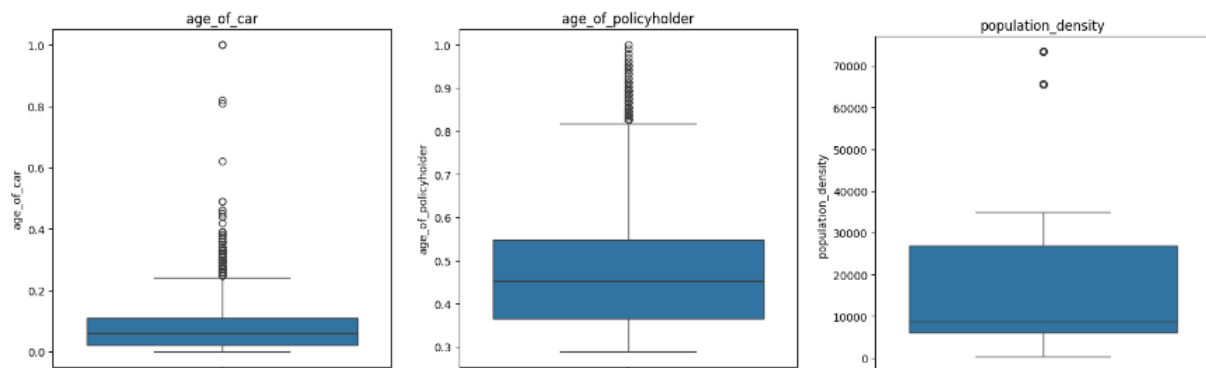


Figure 1.8: Box plots of Numerical Attributes

2.3.2.2 Categorical Attribute Analysis:

A count plot is a bar plot which counts the occurrences of unique values in the attributes. Count plots were plotted for all of the categorical attributes. While most of the categorical attributes had normal distribution with regular values, a couple of columns were having numerical values mixed with characters and measurements. One column is `max_torque` where the values are categorical but are written as “60Nm@3500rpm” (Figure 1.9). This information can be split and extracted into Nm and rpm columns. This way the model can benefit with the extra information provided to it.

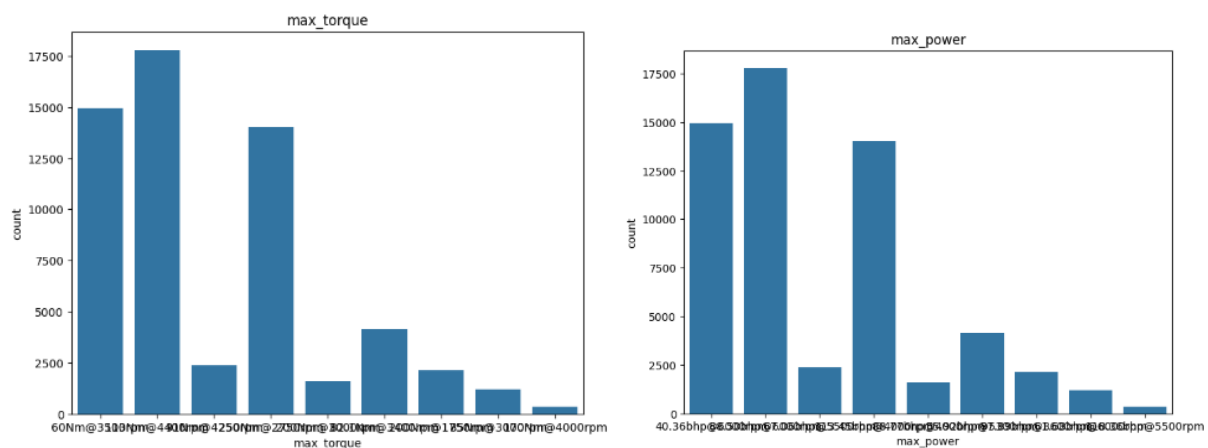


Figure 1.9: Count Plots for `max_torque` and `max_power` columns

2.3.3 Data preprocessing

Outliers found in the numerical columns were dropped based on the threshold set (Figure 2.0).

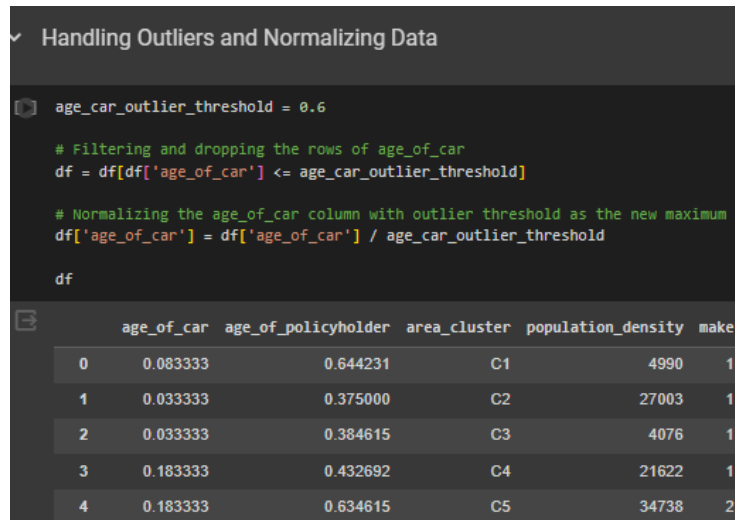


Figure 2.0: Outlier Handling

Similarly, to counter the uneven distribution of data in the `population_density` column, the data is normalized as we can see in Figure 2.1. The data of `population_density` lies between 0 and 1 now.

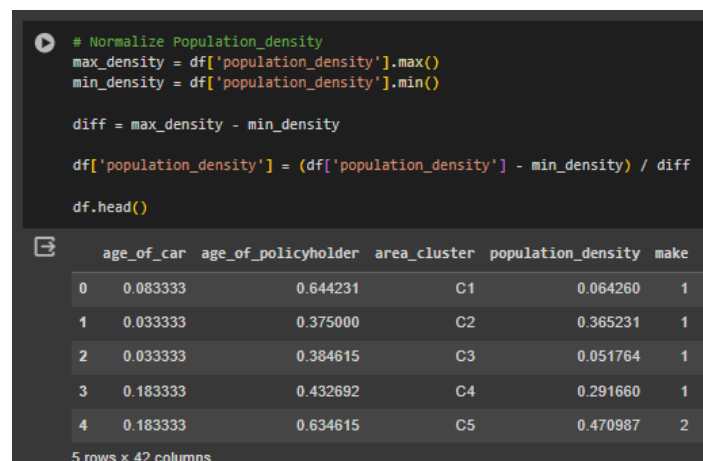


Figure 2.1: Normalized Population Density

Moving on to the categorical attributes, the numbers from `max_powe` and `max_torque` were extracted and added into their separate columns. We can see these were added at the end of the dataframe in Figure 2.2

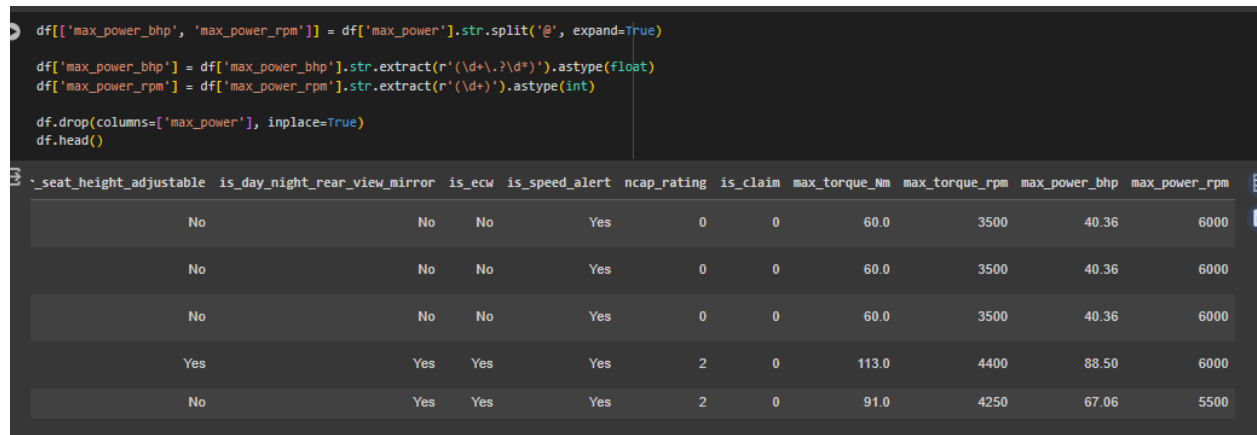


Figure 2.2: New max_power and max_torque columns.

Finally, the labels of categorical columns were encoded with LabelEncoding and OneHotEncoding. The resultant data frame is displayed in Figure 2.3.

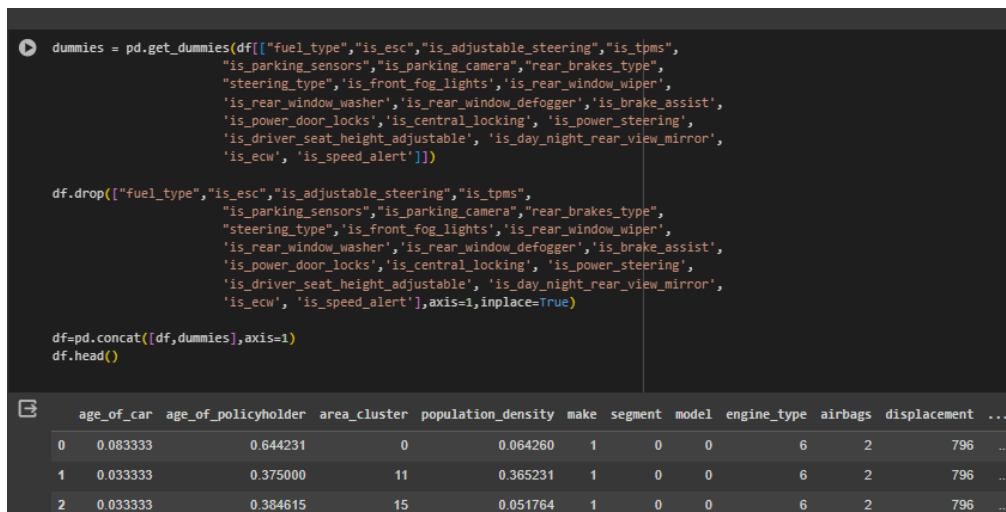


Figure 2.3: Label Encoding of Categorical Attributes

Finally, a Heatmap is generated for all the attributes based on the correlation with each other. The higher the correlation, the more red its respective box will look. We look for very high correlated attributes and remove them as they don't contribute much to the target attributes.

2.3.4 Predictive Modeling

After the train and test datasets are generated, we created predictive models with the popular classifiers such as Logistic Regression [6], Bernoulli Naive Bayesian [7], K Nearest Neighbors Classifier [8], Multi-layer Perceptron Classifier [9], Decision Tree Classifier [10], Random Forest Classifier [11],

AdaBoost Classifier [12], XGBoost Classifier [13], Gradient Boosting Classifier [14], Light Gradient Boosting Classifier [15], Cat Boost Classifier [16]. All of these models were created with their default parameters (Figure 2.4).

```
# Let's go with straightforward modeling, without any parameter tuning.

X = df.drop("is_claim",axis=1)
y = df["is_claim"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

# Classifier Models used

# Logistic Regression
logistic_reg = LogisticRegression().fit(X_train,y_train)

# Bernoulli naive bayesian
naive_bayes = BernoulliNB().fit(X_train,y_train)

# K Nearest Neighbors
k_nearest = KNeighborsClassifier().fit(X_train,y_train)

# Multi-layer Perceptron Classifier
mlp_nn = MLPClassifier().fit(X_train,y_train)
```

Figure 2.4: Generating Classifier Models

Once these models finished training with default parameters, we used evaluation metrics like Precision, Recall, F1-Score, Support and Accuracy. We achieved this using the `classification_report()` method and pretty printing the results (See Figure 2.5).

```
def model_result(model_dict):
    for model_name, model_value in model_dict.items():

        model_accuracy, model_report = get_model_accuracy(model_name, model_value)

        print("-----")
        print("{} Accuracy : {}".format(model_name, model_accuracy))

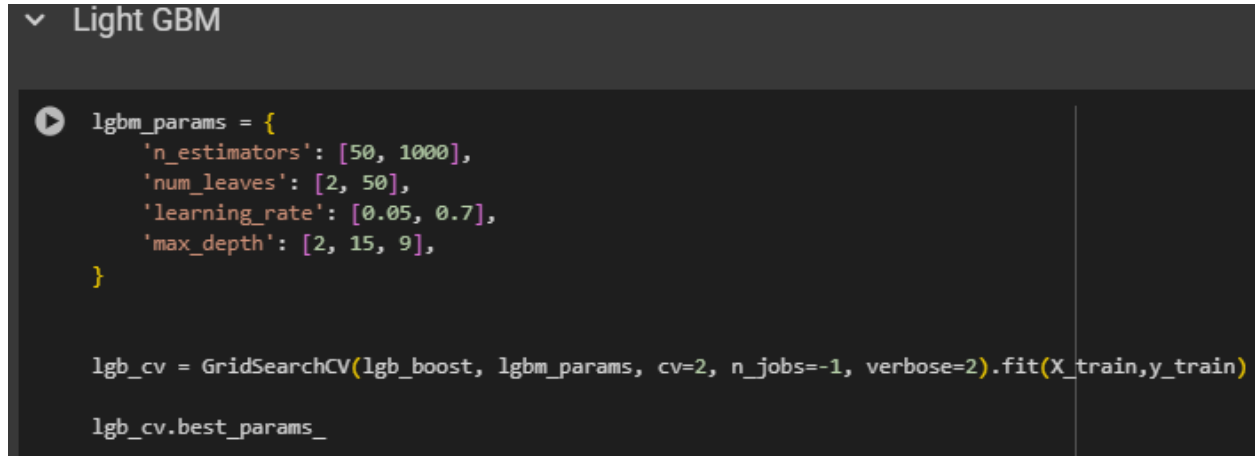
        print()
        print("{} Classification Report: ".format(model_name))
        print(model_report)

    print("*****")
    if(len(model_dict) > 1):
        highest_accuracy = max(model_accuracies.values())
        highest_accuracy_model = [model_name for model_name in model_accuracies if model_accuracies[model_name] == highest_accuracy]
        print("{} Model has the Highest Accuracy: {}".format(highest_accuracy_model, highest_accuracy))
```

Figure 2.5: Printing the Results using Evaluation Metrics

2.3.5 Hyperparameter Tuning:

Based on the results from the previous phase, we have shortlisted the algorithms to a couple of them. On these shortlisted algorithms, we perform hyperparameter tuning to see if we can find a parameter set which has better performance than using the default parameters. This can be done using the GridSearchCV library and OPTUNA Library [17] (Fig 2.6).



```
▼ Light GBM

▶ lgbm_params = {
    'n_estimators': [50, 1000],
    'num_leaves': [2, 50],
    'learning_rate': [0.05, 0.7],
    'max_depth': [2, 15, 9],
}

lgb_cv = GridSearchCV(lgb_boost, lgbm_params, cv=2, n_jobs=-1, verbose=2).fit(X_train,y_train)

lgb_cv.best_params_
```

Figure 2.6 Parameter Grid for GridSearchCV on Light Gradient Boosting Classifier

4. Results:

The last phase is the Model Evaluation phase, where the Model results are evaluated using the metrics described previously. The results can be summarized in Table 1.2 and Table 1.3.

Classifier Model	Final Accuracy
LightGBM	0.9339796545
Multi-layer Perceptron NN	0.9339796545
Logistic Regression	0.9339796545
Naive Bayesian	0.9339796545
AdaBoost	0.9339113812
Gradient Boosting Machine	0.9339113812
XGBoost	0.9332969209
CatBoost	0.9331603741
K Nearest Neighbors	0.9310438998

Random Forest	0.9157506657
Decision Tree	0.8978630436

Table 1.2: Performance of Models on Default Parameters

Table 1.2 shows the initial accuracies of all models when they are trained with the default parameters. We can see that Light Gradient Boosting, Multi-layer perceptron, Logistic Regression, Naive Bayesian algorithms gave an accuracy of 93.39% which is good on its own.

Classifier Model	Final Accuracy
LightGBM	0.9340479279
Multi-layer Perceptron NN	0.9339796545
Logistic Regression	0.9339796545

Table 1.3: Performance of Shortlisted Models after Hyperparameter Tuning

For certain algorithms, such as Logistic Regression, Multi-layer perceptron, Light Gradient Boosting, we tried to improve their accuracy using the GridSearchCV and OPTUNA approaches, but unfortunately, after extensive search and experimentation, no great improvements were found, even for algorithms like Light GBM which was run for almost 300 iterations (Table 1.3) but there was just 0.001% increase in accuracy.

The final best accuracy was achieved by 93.40% for Light Gradient Boosting, The results were satisfactory, and hence no further improvements were pursued.

5. Related Work:

The same problem statement has not been found. But we have found a similar problem statement based on Car Insurance. One of such studies developed a model by authors Ranjodh Singh, Meghna P.Ayyar, Tata Venkata Sri Pavan, Sandeep Gossain, Rajiv Ratan Shah [18] which automates car insurance claims using Deep Learning techniques. They have used network architectures such as Mask R-CNN and PANet which are used for providing exact localization. Whereas training with these architectures also may lack robustness to Real-world variability, its performance in practical applications may be limited.

In another study by the author Patrich Hosien [19], On the Prediction of Automobile Insurance Claims, they have designed a model where, for a given customer, data from past and present customers is used to recommend a better insurance policy that benefits both the customer and the insurance provider. Whereas we developed a model that predicts insurance claims based on policyholder information, which is more useful for insurance companies to identify risky customers who claim their insurance within six months, i.e., before the tenure period, which would be a loss to the insurance companies.

The aim of another study by the author Riki Saito [20], is to examine the viability and efficacy of utilizing learning methodologies to automate and improve several phases of the auto insurance claims procedure.

The study aims to increase efficiency, decrease processing time, and improve customer satisfaction in the auto insurance industry by using deep learning algorithms to expedite decision-making, improve fraud detection, and enhance damage assessment accuracy through image analysis.

6. Conclusion:

In order to plan for the status of insurance claims, we thoroughly analyzed a dataset on motor insurance in this work. We learned a great deal about the variables affecting insurance claims and the efficacy of several machine learning algorithms in predictive modeling through our investigation and modeling work.

Our research showed that a few characteristics, such as the car's age, the kind of engine, and the demographics of the policyholder, had a big influence on how insurance claims turn out. Furthermore, we discovered that some machine learning algorithms performed better than others at accurately forecasting the status of insurance claims, including Random Forest and Gradient Boosting.

Our tests on hyperparameter tuning also showed how crucial it is to optimize model parameters in order to get the greatest results. We were able to reduce overfitting and improve our models' predicted performance by adjusting the model's parameters.

All things considered, our research emphasizes the value of data-driven strategies in the auto insurance sector and the promise of machine learning methods for enhancing risk evaluation and claim forecasting. Our analysis offers valuable information to insurance firms looking to improve claim handling procedures and their predictive modeling skills.

For Future studies, To counter heavy class imbalance scenarios such as this dataset, by implementing techniques like SMOTE Analysis.

References:

- [1] *Iftesha Nanjin, "Car Insurance Claim", Feb 2023, Kaggle.com*
[<https://www.kaggle.com/datasets/ifteshanajnin/carinsuranceclaimprediction-classification>]
- [2] *Analytics Vidhya, "Dataverse Hack Competition", 2022*
[<https://datahack.analyticsvidhya.com/contest/dataverse/True/#About>]
- [3] *Python Software Foundation, "Python Programming language"* [<https://www.python.org/>]
- [4] *Jupyter, Jupyter Notebook* [<https://jupyter.org/>]
- [5] *Google, Google Colaboratory* [<https://colab.research.google.com/>]
- [6] *Scikit-learn, Logistic Regression*
[https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html]

- [7] Scikit-learn, *Bernoulli Naive Bayesian*
[https://scikit-learn.org/stable/modules/naive_bayes.html#bernoulli-naive-bayes]
- [8] Scikit-learn, *K Nearest Neighbors Classifier*
[<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>]
- [9] Scikit-learn, *Multi-layer perceptron Classifier*
[https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html]
- [10] Scikit-learn, *Decision Tree Classifier*
[<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>]
- [11] Scikit-learn, *Random Forest Classifier*
[<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>]
- [12] Scikit-learn, *AdaBoost Classifier*
[<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>]
- [13] xgboost, *XGBoost Classifier* [<https://xgboost.readthedocs.io/en/stable/>]
- [14] Scikit-learn, *Gradient Boosting Classifier*
[<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>]
- [15] Microsoft, *Light Gradient Boosting Classifier*
[<https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html>]
- [16] Cat Boost Classifier [<https://catboost.ai/>]
- [17] Optuna, *Optuna Tuning* [<https://optuna.readthedocs.io/en/stable/>]
- [18] Ranjodh Singh, Meghna P.Ayyar, Tata Venkata Sri Pavan, Sandeep Gossain, Rajiv Ratan Shah,
“Automating Car Insurance claims using Deep learning Techniques”.
[<https://ieeexplore.ieee.org/document/8919258/authors#authors>]
- [19] Patrick Hosein, “On the Prediction of Automobile Insurance Claims”.
[<https://ieeexplore.ieee.org/document/9739635/authors#authors>]
- [20] Riki Saito, “Twin Cities Auto Insurance Rate-Modeling Project”
[<https://github.com/rjsaito/Auto-Insurance-Project>]