**CHAPTER THREE**

**EXPERIMENTAL METHODOLOGY**

The end-to-end machine learning (ML) process for this project encompasses the following technical steps:

1. **Library Setup and Framework Implementation:** Installation and importation of essential libraries, packages, and ML frameworks for data manipulation, analysis, wrangling, and cleansing of the dataset. This includes frameworks for data handling (e.g., Pandas), numerical computations (e.g., NumPy), visualization (e.g., Matplotlib, Seaborn), and ML algorithms (e.g., SciKit-learn).
2. **Statistical Summary and Data Profiling:** Conducting a statistical summary and data profiling to comprehend the dataset's structure, key statistical measures, and initial insights. This involves descriptive statistics, identifying missing values, distributions, and basic visualizations to understand data characteristics.
3. **Data Cleaning and Pre-processing:** Implementing data cleaning procedures, including handling missing values, addressing duplicates, and resolving inconsistencies. Pre-processing steps involve feature scaling, encoding categorical variables, handling outliers, and transforming data for ML model compatibility.
4. **Exploratory Data Analysis (EDA):** Performing EDA to uncover patterns, relationships, and insights within the dataset. This involves visualizations, correlation analysis, and identifying relevant features for modelling.
5. **Basic Modelling and Pipeline Construction:** Creating a foundational ML pipeline by implementing initial classification models using standard algorithms (e.g., Decision Trees, Logistic Regression) to establish a baseline performance.
6. **Evaluation of Classification Models:** Evaluating the performance of basic classification models using appropriate metrics (e.g., accuracy, precision, recall, F1-score) and techniques like cross-validation to assess model robustness.
7. **Feature Engineering and Mapping:** Conducting feature engineering to create new informative features and mapping techniques for categorical variables to enhance model performance and interpretability.
8. **Ensemble Learning with Voting Classifier:** Implementing ensemble learning using a Voting Classifier to combine multiple models for improved predictive performance.
9. **Feature Importance Analysis:** Determining feature importance through techniques like permutation importance or feature contribution analysis to understand influential variables in model predictions.
10. **Model Saving and Deployment:** Saving the finalized ML model for future use or deployment in production environments to make predictions on new data.

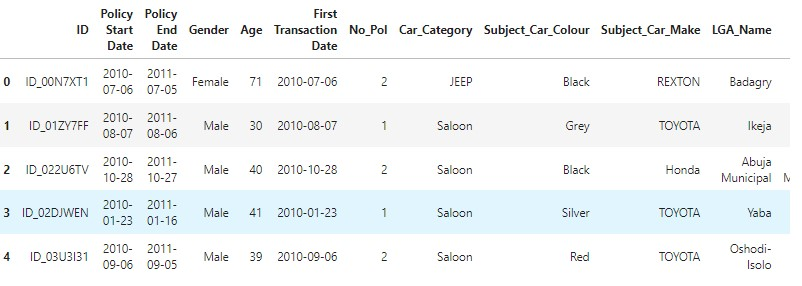
**3.1 Library Setup and Framework Implementation**

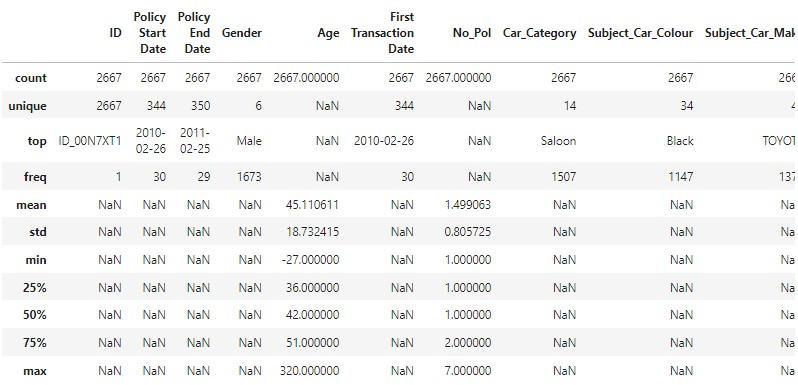
The entire process was executed using the Python programming language. The table provided outlines the diverse packages and modules utilized at each stage of the process.

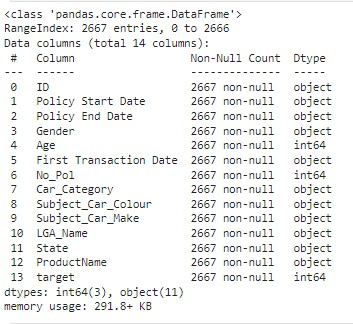
|  |  |  |
| --- | --- | --- |
| Stage | Packages/Modules | Submodules/Class Functions |
| Data Exploration & Analysis | Pandas | DataFrame, Series |
|  | Sweet viz | sv.analyze() (analyze), sv.compare() (compare) |
|  | Dtale | dt.show() (show) |
|  | Seaborn | Various visualization functions |
|  | Matplotlib | plt (plotting functions), plb (plotting), gridspec (plotting) |
|  | NumPy | np (numerical operations) |
| Data Pre-processing | Fuzzywuzzy | process (fuzzy matching) |
|  | Pandas and NumPy | Basic data manipulation |
| Imbalanced Data Handling | Imblearn | RandomOverSampler (oversampling technique for handling class imbalance) |
| Model Building and Evaluation | SciKit-Learn | Pre-processing: SimpleImputer, LabelEncoder, StandardScaler, OneHotEncoder, RobustScaler, OrdinalEncoder Model Selection: LogisticRegression, RandomForestClassifier, SVC, etc. |
|  |  | Model Evaluation: accuracy\_score, confusion\_matrix, classification\_report, roc\_curve Pipeline & Compose: Pipeline, ColumnTransformer, make\_pipeline |
|  | Lazy predict | LazyClassifier (quick model selection) |
|  | Category\_encoders | OrdinalEncoder (categorical encoding) |
|  | Joblib | Joblib. dump () (model saving) |

**3.2** **Statistical Summary and Data Profiling**

Sweetviz and Dtale were employed to generate a comprehensive data profile for the dataset, compiling statistical insights, visualizations, and interactive summaries. This detailed profile was then exported and saved as an HTML file, encapsulating the dataset's descriptive statistics, distributions, missing values, correlations, and graphical representations for in-depth analysis and future reference.







**3.3 Data Cleaning and Pre-processing**

Data cleaning and pre-processing involved actions aimed at refining the dataset to facilitate successful model fitting. The measures employed involve handling missing values, addressing outliers, standardizing or normalizing features, and encoding categorical variables. Other essential steps: firstly, scaling features to ensure uniformity in their ranges, preventing bias towards certain features during training. Secondly, handling categorical data through encoding techniques like one-hot encoding, enabling algorithms to process categorical information accurately. Thirdly, addressing imbalanced datasets by employing oversampling, under sampling, or specialized algorithms to counteract label distribution discrepancies.

**3.3.1 Addressing Missing and Duplicated Values**

The initial stage of data cleaning involved employing the Pandas function to identify and handle missing values within the dataset. Upon inspection, it was found that none of the feature columns contained any missing values, ensuring completeness in the dataset. Additionally, the duplicate check utilizing Pandas returned a False result, indicating the absence of any duplicated entries within the dataset. These measures confirmed the dataset's integrity, free from missing values and duplicate records.

**3.3.2 Handling Anomalies**

Upon the initial assessment of the 'Age' feature, anomalies surfaced, including negative values and excessively high entries. To rectify these issues, the absolute values were taken to address negative values, while outliers were replaced with more plausible values within an acceptable range. Additionally, discrepancies were observed in the allocation of certain LGAs (Local Government Areas) to their respective states. For instance, 'Badagry' was incorrectly placed within 'Benue,' potentially impeding our analysis progress. To mitigate this, a mapping process was initiated, utilizing a domain list sourced from reputable internet resources. This list contained accurate state keys and corresponding LGAs, enabling the correction of mismatches and ensuring accurate allocation of LGAs to their respective states for a more reliable analysis.

**3.3.3 Standardizing Categorical Data**

In order to standardize our categorical data, an examination of the subject car color column revealed multiple diverse and ambiguous categories, potentially complicating understanding for our ML algorithm. To streamline and simplify this classification for better algorithmic comprehension, the labels were transformed into a more straightforward set, encompassing 'Black', 'Grey', 'Silver', 'Red', 'Green', 'Ash', 'Blue', 'White', 'Brown', 'Gold', 'Purple', and 'Yellow'. This reclassification aimed to reduce complexity and enhance the clarity of categorical data for improved model interpretation and performance.

In an effort to standardize the 'Gender' column, it was noted that several non-feasible and varied categories such as 'Entity', 'NO GENDER', 'Joint Gender', and 'NOT STATED' were present alongside 'Female' and 'Male'. To address this inconsistency and create a more uniform representation, a dictionary mapping specifically 'Male' and 'Female' label was applied to generate a new column comprising only these two distinct categories. This standardization aimed to streamline the 'Gender' data, enhancing its clarity and usability for subsequent analysis and modelling purposes.

**3.3.4 Standardizing Local Government Area Names Using 'StateName’**

Upon investigation, inconsistencies were identified in the nomenclature of local government areas ('LGA\_Name'). To rectify these discrepancies and ensure consistency across the dataset, we utilized information from the 'StateName' dataset. Misspelled or inconsistently labelled entries in the 'LGA\_Name' column were corrected by referencing the 'StateName' dataset. This process aimed to standardize the 'LGA\_Name' entries, ensuring uniformity and accuracy within the dataset for more reliable analyses and modelling.

**3.4 Explorative Data Analysis**

Exploratory Data Analysis (EDA) is a crucial preliminary step in data analysis that involves examining and visualizing datasets to comprehend their structures, patterns, and potential relationships between variables. EDA serves as a foundation for further analysis, aiding in uncovering insights and informing subsequent decision-making processes within data-driven tasks.

The Procedural steps in the Explorative Data Analysis;

1. Data Profiling and Summary:
2. Initial inspection of dataset dimensions, structure, and data types.
3. Generation of summary statistics (mean, median, standard deviation) and data profiling to understand distributions and central tendencies.
4. Univariate Analysis:
5. Examination of individual variables/features in isolation to comprehend their distributions, using histograms, bar charts, or box plots.
6. Bivariate Analysis:
7. Investigation of relationships between pairs of variables to identify potential correlations or dependencies using scatter plots, heatmaps, or correlation matrices.
8. Multivariate Analysis:
9. Exploration of interactions among multiple variables to understand complex relationships using techniques like dimensionality reduction or cluster analysis.
10. Visualization Techniques:
11. Usage of visual aids like histograms, scatter plots, pie charts, or heatmaps to represent data distributions, trends, and patterns effectively.

Exploratory Data Analysis plays a pivotal role in understanding data characteristics, uncovering insights, and guiding subsequent steps in data pre-processing, feature engineering, and model selection within the data analysis pipeline. This systematic approach helps in gaining valuable insights into the dataset, paving the way for informed decision-making and robust model building.

**3.5 Basic Modelling and Pipeline Construction**

This entails the creation of predictive models and pipelines using various machine learning algorithms for classification tasks on a dataset. Here's a summary of the processes involved, models utilized, metrics evaluated, and notable hyperparameters:

1. Data Splitting: The dataset is divided into training (80%) and test sets (20%) to train models and evaluate their performance.
2. Feature Selection and Label Definition:

* Features: Identified features such as 'Gender', 'Age', 'First Transaction Date', 'No\_Pol', 'Car\_Category', 'Subject\_Car\_Colour', 'Subject\_Car\_Make', 'LGA\_Name', 'State', and 'ProductName'.
* Target Variable: Defined 'target' as the prediction label.

1. Pre-processing Pipeline:

* Numerical Features: Scaling performed using StandardScaler.
* Categorical Features: Encoding with OneHotEncoder to convert categorical data into a format suitable for machine learning models.

1. Model Building and Training:

* Logistic Regression: A classification model used to predict the target variable.
* Decision Tree Classifier: Constructed a decision tree-based model for classification.
* K-Nearest Neighbours (KNN) Classifier: Utilized for classification based on nearest neighbours.
* Support Vector Machine (SVM) Classifier: Employed a kernel-based model for classification.
* Gradient Boosting Classifier: Constructed an ensemble model using boosting techniques.

**3.6 Evaluation of Classification Models**

**3.6.1 Accuracy Score:**

* Definition: Accuracy measures the ratio of correctly predicted instances to the total instances in the dataset.
* Calculation: Accuracy = (Number of Correct Predictions)/(Total Number of Predictions)×100
* Use: It's a common metric for evaluating classification models. However, in imbalanced datasets, where one class dominates the others, accuracy might not be the ideal metric as it can be misleading.

**3.6.2 Classification Report:**

Components:

* Precision: The ratio of correctly predicted positive observations to the total predicted positive observations.
* Recall (Sensitivity): The ratio of correctly predicted positive observations to the all actual positives.
* F1-Score: The harmonic mean of precision and recall, offering a balance between the two metrics.
* Support: The number of occurrences of each class in the dataset.

Use: Provides a detailed analysis of a model's performance for individual classes. It's beneficial when class imbalance exists in the dataset.

**3.6.3 Confusion Matrix:**

Components:

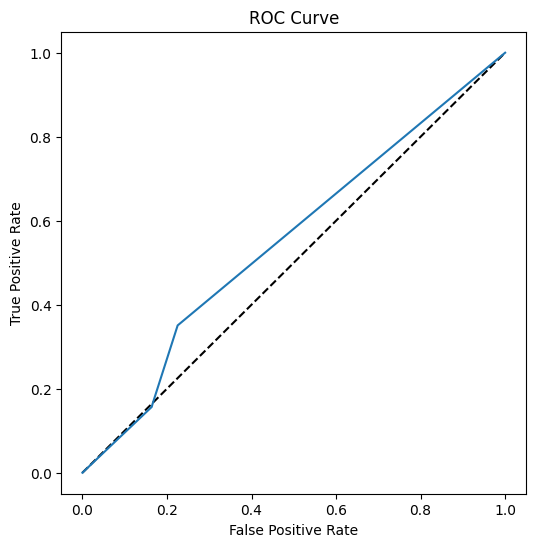
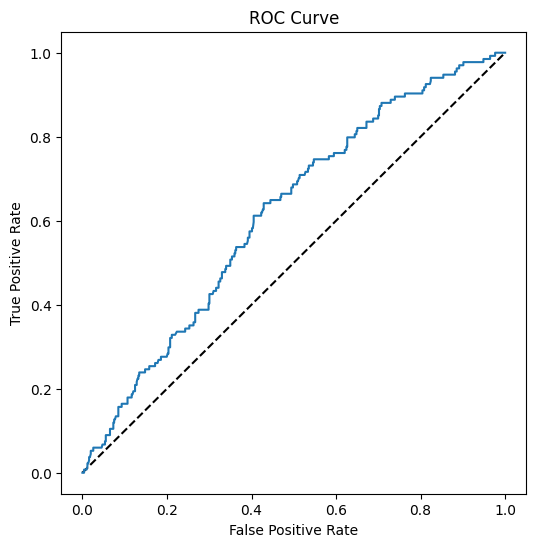
* True Positive (TP): Instances correctly predicted as positive.
* True Negative (TN): Instances correctly predicted as negative.
* False Positive (FP): Instances incorrectly predicted as positive.
* False Negative (FN): Instances incorrectly predicted as negative.

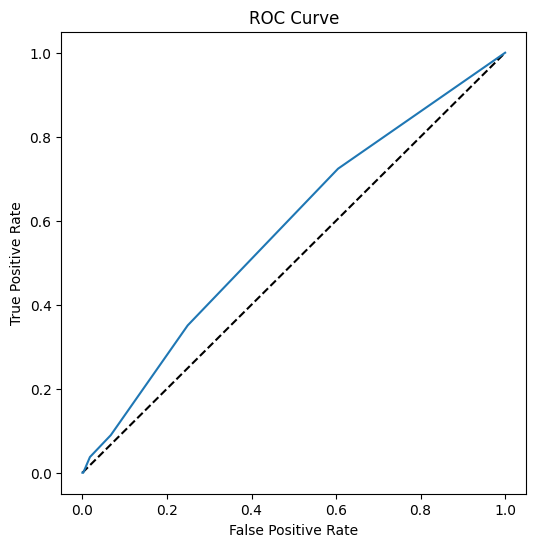
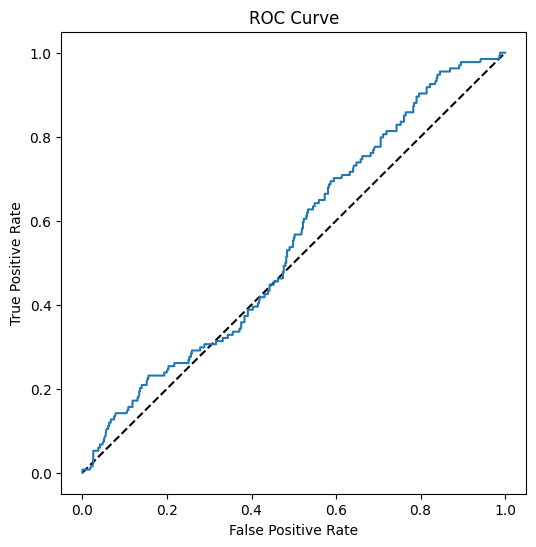
Use: Offers a visual representation of a model's performance and helps assess its predictive accuracy, especially in understanding misclassifications.

**3.6.4 ROC Curves (Receiver Operating Characteristic Curves):**

* Definition: ROC curves illustrate the performance of a classification model at various threshold settings.
* X-axis: False Positive Rate (FPR) = (False Positives)/(False Positives +True Negatives)
* Y-axis: True Positive Rate (TPR) or Sensitivity = (True Positives)/(True Positives+ False Negatives)
* Use: Evaluates the trade-off between sensitivity and specificity, aiding in threshold selection for optimal model performance.

These metrics collectively provide a comprehensive understanding of a model's predictive capabilities, highlighting its strengths and weaknesses across various aspects of classification tasks. They enable practitioners to make informed decisions regarding model selection, fine-tuning, and optimization for better performance.



**3.7 Feature Engineering and Mapping**

Feature engineering is a crucial phase in machine learning where new features are created or modified from existing ones to enhance model performance and capture essential information from the dataset. In this context, the process involved various transformations and creations of features based on temporal, categorical, and numerical attributes. These engineered features aim to provide richer information, better representation, and more predictive power for the models in use. The procedures implemented involved transforming date features into numerical representations, segmenting ages into categories, deriving new features from existing ones, and converting categorical variables into suitable formats for machine learning algorithms.

Here's a step-by-step overview of the feature engineering process you've implemented:

**3.7.1 Date Conversion:**

* Converted 'Policy Start Date' and 'Policy End Date' columns to datetime format using pd.to\_datetime ().

**3.7.2 Policy Duration Calculation:**

* Created a new feature 'Policy Duration' by calculating the number of days between 'Policy End Date' and 'Policy Start Date'.

**3.7.3 Extracting Date Components:**

* Extracted year, month, and day of the week from 'Policy Start Date' into separate columns ('Policy Start Year', 'Policy Start Month', 'Policy Start Day of Week').

**3.7.4 Age Categorization:**

* Segmented ages into categories ('Child', 'Young Adult', 'Adult', 'Senior') using predefined bins and labels via pd.cut ().

**3.7.5 Days Since First Transaction:**

* Calculated the number of days between 'Policy Start Date' and 'First Transaction Date' and stored it as 'Days Since First Transaction'.

**3.7.6 One-Hot Encoding:**

* Applied one-hot encoding to categorical columns ('ProductName') using pd.get\_dummies () to convert them into numerical format.

**3.7.7 Label Encoding:**

* Encoded 'Gender' column using LabelEncoder () to convert categorical values into numerical representation.

**3.7.8 Feature Interaction:**

* Created a new feature 'Age\_No\_Pol' by multiplying 'Age' and 'No\_Pol' columns to capture the interaction between these features.

**3.7.9 Period Conversion:**

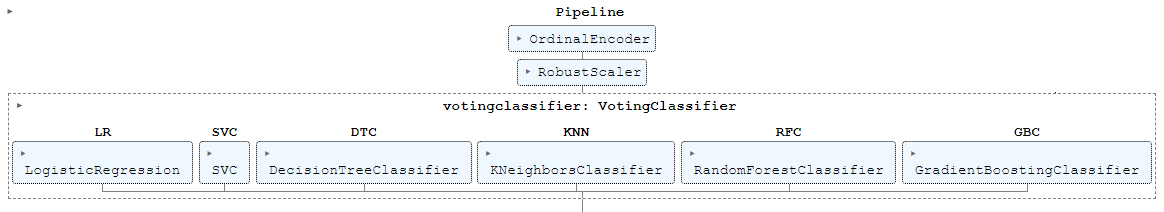
* Converted date features ('Policy Start Date', 'Policy End Date', 'First Transaction Date') into the number of days since a certain date (January 1, 1970) using timestamp conversion.

**3.8 Ensemble Learning and Voting Classifier**

This section covers the process of building and evaluating a robust machine learning model using ensemble techniques and resampling methods. The provided code demonstrates the construction of a robust model and subsequent evaluation metrics.

**3.8.1 Data Splitting and Resampling:**

* split\_data () Function: Splits the data into training and test sets, dropping irrelevant columns. Returns X\_train, X\_test, y\_train, and y\_test.
* resample () Function: Performs oversampling on the training data to address class imbalance using RandomOverSampler.
  + 1. **Model Building**:
* build model () Function: Constructs a Voting Classifier ensemble model with multiple base estimators (Logistic Regression, SVC, Decision Tree, KNN, Random Forest, Gradient Boosting).



**3.8.3 Model Training and Evaluation:**

* TOAD model Training: Utilizes build model () to create a model, oversamples the training data using resample (), and fits the model on the oversampled data.
* Model Evaluation: Prints the accuracy scores for both the test and train data, classification report showing precision, recall, and F1-score, and compares the feature names and columns to ensure consistency.

**3.8.4 Model Analysis:**

* Feature Importance: Attempts to extract feature importances from the model (if available) but might require correction in the code as TOADmodel may not possess such attributes.

**3.8.5 Model Structure and Parameters**:

* Demonstrates the structure of the constructed model using the named steps attribute to access individual steps in the pipeline.

**CHAPTER FOUR**

**RESULTS/APPLICATIONS AND DISCUSSION**

**4.0 Introduction**

In this chapter, our primary aim is to provide a comprehensive walkthrough of the data analysis process and the subsequent results obtained from the modelling phase, specifically addressing the auto insurance initiative. Our approach encompasses detailed exploratory data analysis techniques—such as univariate, bivariate, and multivariate analyses—coupled with thorough data pre-processing and machine learning methodologies. The overarching goal is to present a complete overview of our findings, particularly in establishing and evaluating a robust predictive classification framework. This framework intends to forecast target labels (Class 0 and 1), offering insights into whether a customer is likely to file an insurance claim.

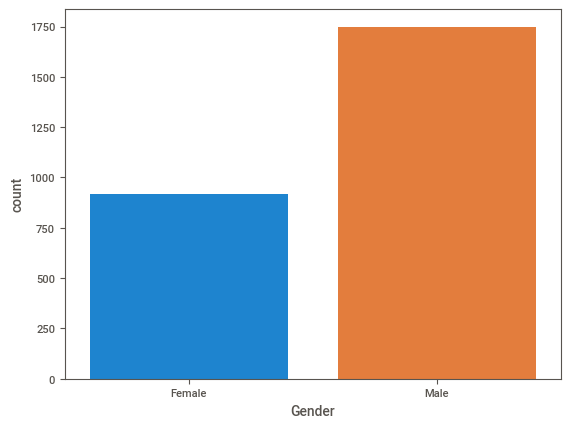
**4.1 Exploratory Data Analysis**

Exploratory Data Analysis (EDA) holds significant importance in machine learning as it allows for a comprehensive understanding of the dataset's characteristics, patterns, and relationships. EDA aids in familiarizing oneself with the data, uncovering underlying stories, and revealing insights crucial for informed decision-making during the modelling process. Through EDA, one can identify outliers, missing values, distributions, correlations between variables, and potential interactions among features, thereby facilitating data pre-processing and feature selection. This initial exploration serves as a foundation for constructing robust models and making informed choices about feature engineering or model selection based on a deeper comprehension of the dataset's nuances and behaviours.

To gain a comprehensive understanding of our dataset and unearth its underlying narrative, an initial data profiling was conducted. This entailed univariate analysis for each column, utilizing bar charts and distribution plots to visualize individual feature distributions. Bivariate analysis was employed to examine correlations and relationships between features and target columns. Specifically, this involved assessing how different features relate to the target variable. Furthermore, to extract deeper insights and explore more complex relationships within the data, multivariate analysis was performed. This comprehensive approach aimed to unravel various dimensions of the dataset, from individual feature distributions to interdependencies among multiple features and their impact on the target variable.

**4.2.1 Univariate Analysis**

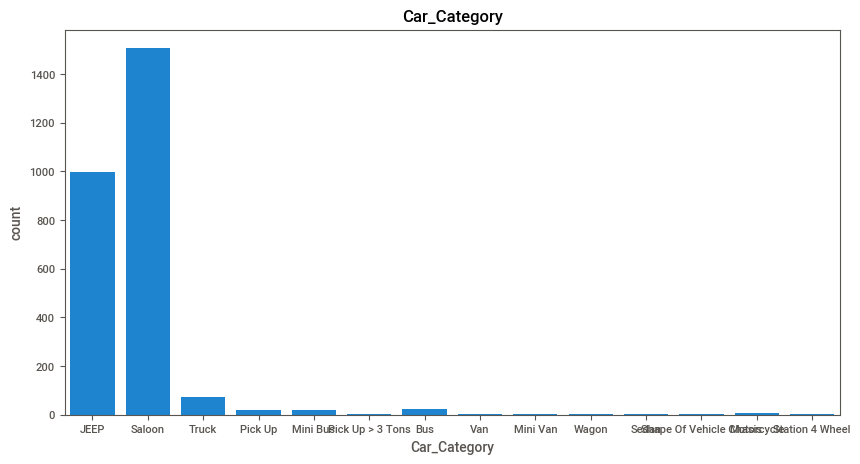
The examination of gender distribution within the dataset uncovers a slight predominance of females (53%) in comparison to males (47%). Despite this narrow margin, statistical significance is observed. The data indicates a gender ratio of 0.86, implying the presence of 86 males for every 100 females. Moreover, the 95% confidence interval for the disparity in gender proportions spans from 2% to 12%, suggesting that the actual difference in gender representation is likely to fall within this range with a high level of confidence.



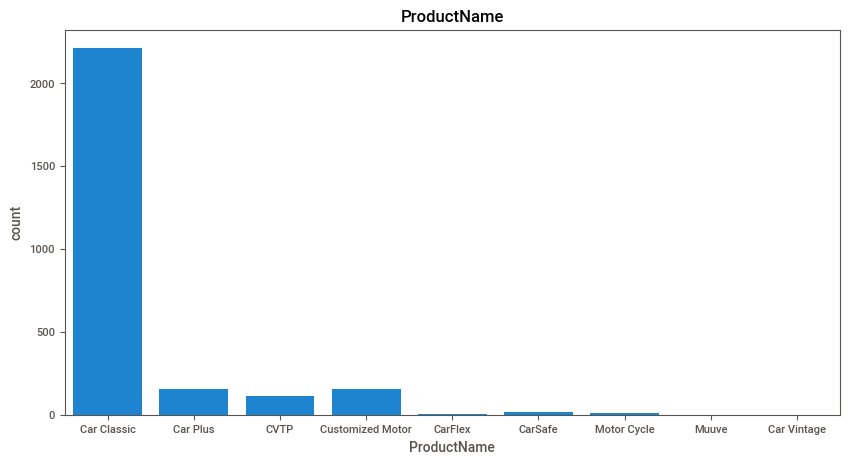
The bar graph depicting Nigeria's car distribution illustrates a predominant preference for Saloon, Truck, and Pick vehicles, collectively constituting over 70% of the total. In contrast, Mini, Valastraßeren, and Vellotated Wheel cars represent the least favoured options.

This preference for larger vehicles might be attributed to factors such as family size, transportation needs for goods, considerations related to terrain, or potential pricing differences. The higher cost associated with Saloons, Trucks, and Picks compared to Minis and Valastraßeren suggests the presence of potentially higher disposable income among Nigerians.

The Nigerian automotive market appears dynamic, impacting the demand for different car types. However, uncertainties persist around the categories "Valastraßeren" and "Vellotated Wheel," possibly indicating misspellings or specific Nigerian car types. While the graph doesn't specify units, it likely represents the quantity of cars in each category.

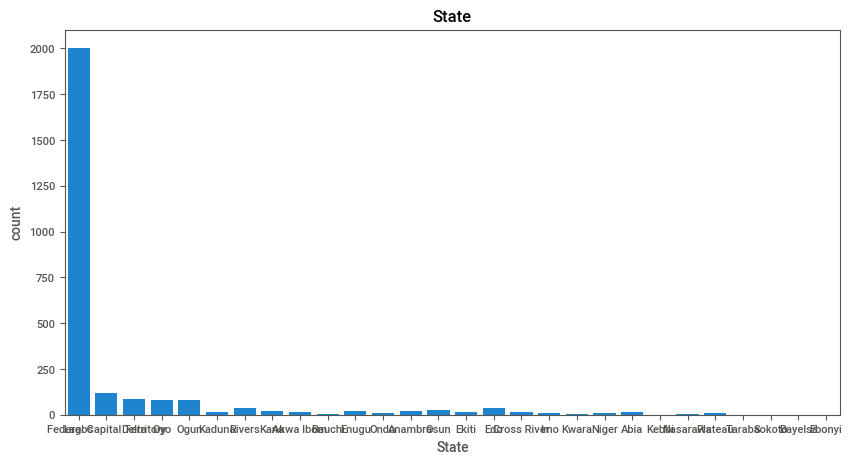
The dataset presents the distribution of product names across various car types, with "car classic" emerging as the most prevalent, followed by "car plus," "car voltage," and "customized motor." Less common types include "car vintage" and "muuve."

This diverse range signifies a wide array of car types available to accommodate distinct preferences among consumers. The popularity of "car classic" and "car plus" indicates a demand for both traditional and contemporary vehicles. Additionally, the prominence of "car voltage" reflects the increasing presence of electric vehicles in the market.



The visualization illustrates the distribution of states within the automobile industry, highlighting prominent states like Lagos, Ogun, Oyo, Delta, and Edo, alongside less prominent ones such as Yobe, Taraba, Ebonyi, Gombe, and Adamawa.

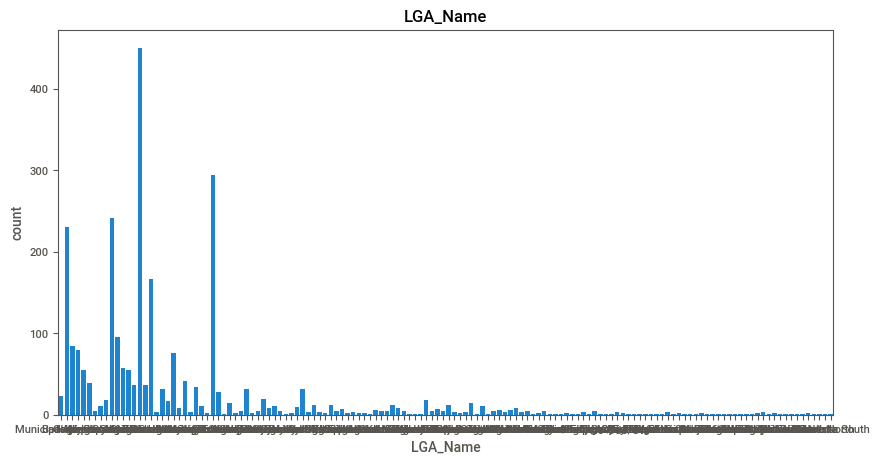
This concentration suggests focal points in Lagos, Ogun, and Oyo, renowned for their industrial and commercial significance. Additionally, the presence of oil and gas resources in Delta and Edo contributes to their importance within the automobile sector.

The image depicts the Local Government Area (LGA) distribution within the automobile dataset, highlighting prominent LGAs like Alimosho, Surulere, Ikeja, Oshodi-Isolo, and Eti-Osa, alongside less prevalent ones such as Gwagwalada, Kuje, Abaji, Bwari, and Kwali.

This concentration underscores the significance of key LGAs within Lagos State, renowned as commercial and industrial hubs housing numerous automobile manufacturers and dealerships.

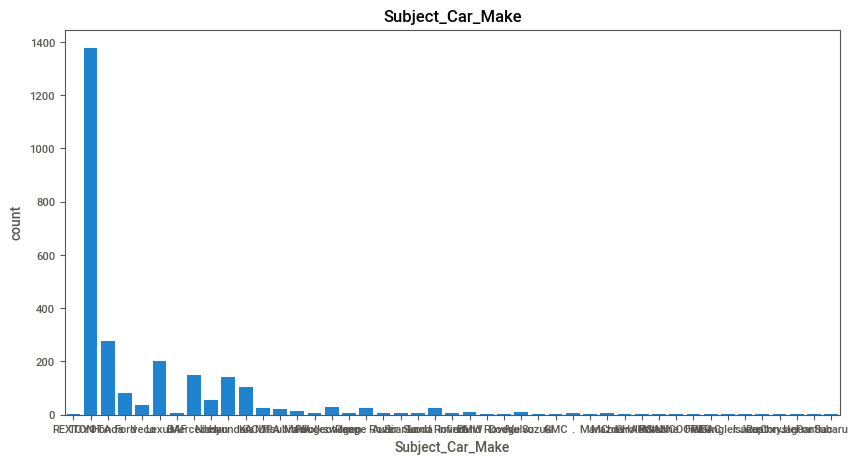
Several contributing factors shape this concentration, including superior infrastructure facilitating the transportation of cars and components, as well as a substantial skilled workforce pivotal to the automobile industry's operations.

For consumers, this concentration in specific LGAs translates to increased accessibility to cars and related services. However, individuals in other LGAs might encounter elevated costs or the necessity to travel longer distances for their car-related needs.



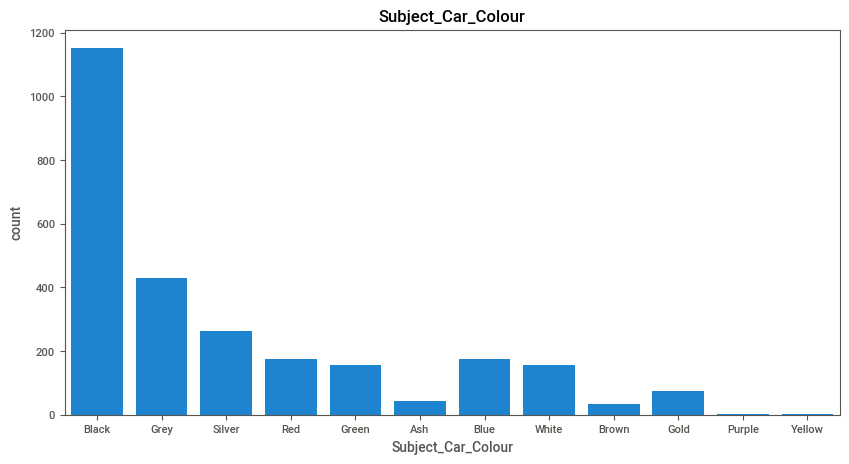
The distribution of Subject\_Car\_Make in Nigeria showcases a preference for Toyota, Honda, Nissan, Hyundai, and Kia, while Land Rover, Audi, Mercedes-Benz, BMW, and Ford hold a smaller presence.

This dominance of Japanese car makes can be attributed to their perceived reliability and affordability, with Korean brands also gaining traction for their value-for-money offerings. This pattern reflects Nigerian consumers' strong brand loyalty.

****

The Subject\_Car\_Colour distribution in Nigeria reveals Black as the most prevalent, succeeded by White, Grey, Silver, and Blue, while Gold, Brown, Green, Yellow, and Purple have lower popularity.

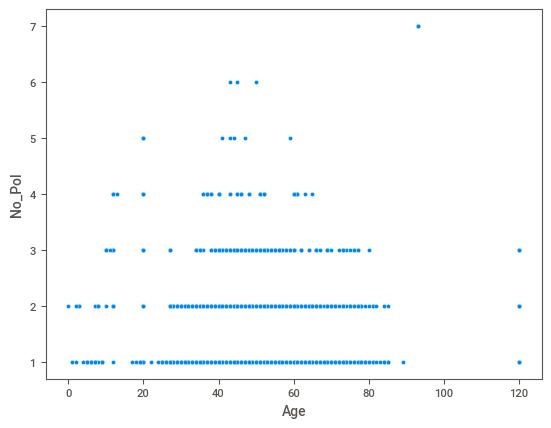
This trend indicates a penchant for neutral car colours such as Black, White, Grey, and Silver, likely attributed to their perceived sophistication, ease of maintenance, or potentially higher resale value.

****

**4.2.2 Bivariate Analysis**

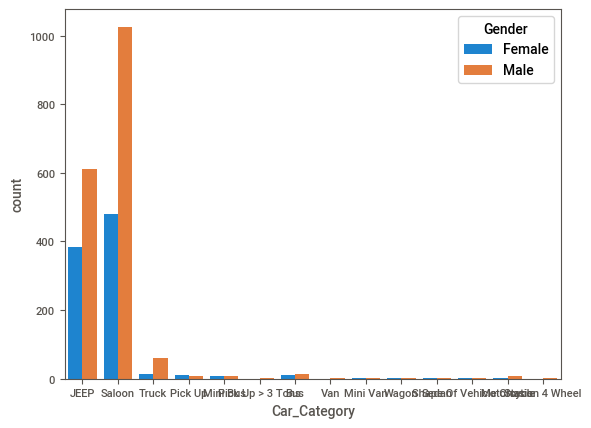
To explore the dataset's feature relationships, the analysis focused on several key questions:

1. Is there a correlation between the policy holder's age and the quantity of policies they possess?
2. Does gender influence the type of car owned?
3. Do individuals in distinct states differ in the number of policies they hold?
4. Is there a connection between the car make and the target variable?
5. Does the product name exert an impact on the target variable?
6. What is the average age of policy holders within the most prevalent car category among those with a higher policy count, specifically among male policy holders?

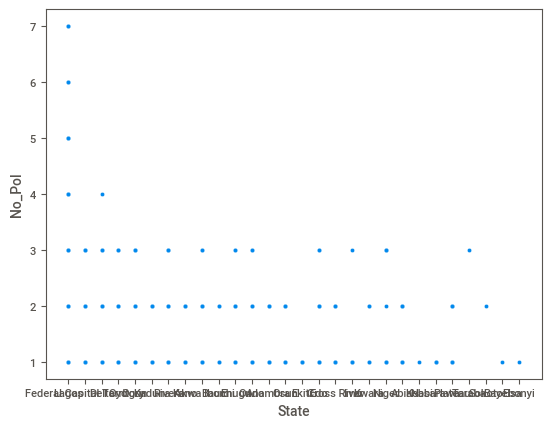
****

Based on the presented visualization, no distinct correlation is evident between the age of policy holders and the quantity of policies they possess. The data showcases a broad spectrum of ages across varying policy counts, lacking a discernible trend or pattern.

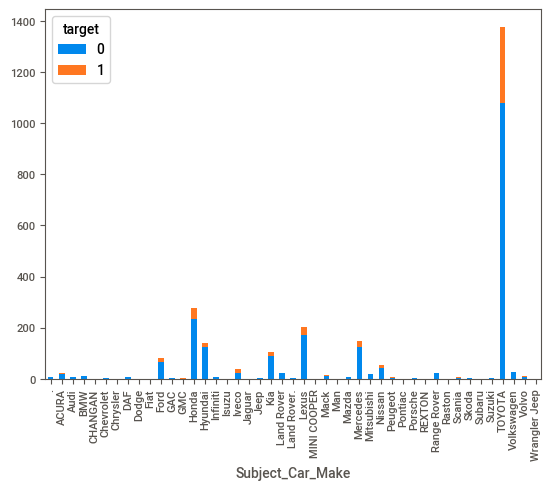
For instance, individuals in their 20s may hold multiple policies, while others in their 60s may possess just one policy. Similarly, individuals in their 40s might hold multiple policies, contrasting with those in their 70s with singular policy holdings.

****

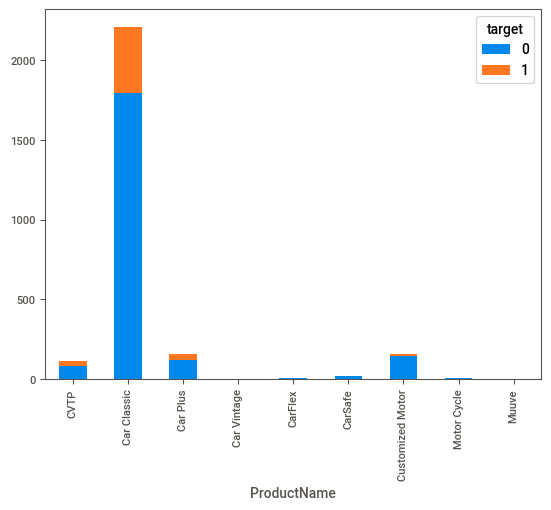
The visualization depicts the ownership percentage of Jeep pickup trucks categorized by gender. It indicates a higher ownership percentage among men compared to women for this specific vehicle type, implying a potential association between gender and the ownership of Jeep pickup trucks.

****

No significant discrepancy exists in the number of policies held by individuals across various states, as indicated in the provided image. The distribution of policies held by individuals in distinct states appears notably alike. Each state showcases a median policy count of 2, and the interquartile range (IQR) demonstrates a similarity across states, primarily around 1.5.



The distribution provided illustrates the count of targets associated with each car make. Toyota emerges with the highest number of targets, followed by Honda, Nissan, Hyundai, and Kia. Conversely, Land Rover holds the lowest count, succeeded by Audi, Mercedes-Benz, BMW, and Ford.



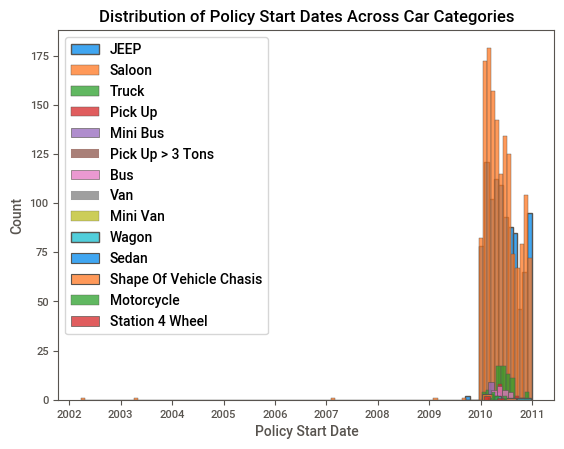
The distribution of targets by product name reveals Car Classic as the product name with the highest target count, followed by Car Plus, Car Vintage, CarFlex, and CarSafe. Conversely, Customized Motor holds the lowest target count, followed by Motor Cycle, Muuve, Vellotated Wheel, and Valastraßeren.

However, this data lacks sufficient evidence to conclusively establish the influence of the product name on the target variable. The target count attributed to each product name could be affected by several factors, including product name popularity, the product's type, and the nature of the target variable itself.

For instance, the high target count associated with Car Classic aligns with its popularity and broad applicability as a general-purpose car, appealing to a wide audience. Conversely, the lower count linked to Customized Motor, being a less popular and potentially specialized product, might attract a more niche market. Moreover, its higher cost could diminish its attractiveness to certain consumers. These factors illustrate how diverse elements beyond the product name may impact the count of targets related to each product.

**4.2.3 Multivariate Analysis**

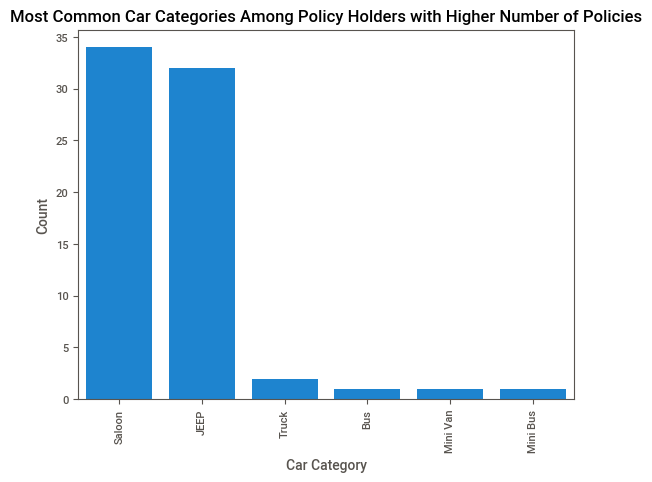
**1. What is the distribution of policy start date across different car categories?**

****

The distribution of policy start date across different car categories shows that SUVs and sedans are the most popular car categories, with a higher number of policies started in the past few years. Pickups and trucks are also popular, with a more gradual increase in the number of policies started over time. Vans and minivans are less popular, with a smaller number of policies started overall.

* SUVs and sedans are more popular car categories overall, so there is a naturally higher number of policies started for these categories.
* SUVs and sedans are typically newer cars, so they may be more likely to be insured.
* Pickups and trucks are often used for work or commercial purposes, so they may be less likely to be insured.
* Vans and minivans are less popular car categories overall, so there is a naturally lower number of policies started for these categories.
* Vans and minivans are often older cars, so they may be less likely to be insured.

**2. What is the most common car category among the car category among policy holders with a higher number of policies**

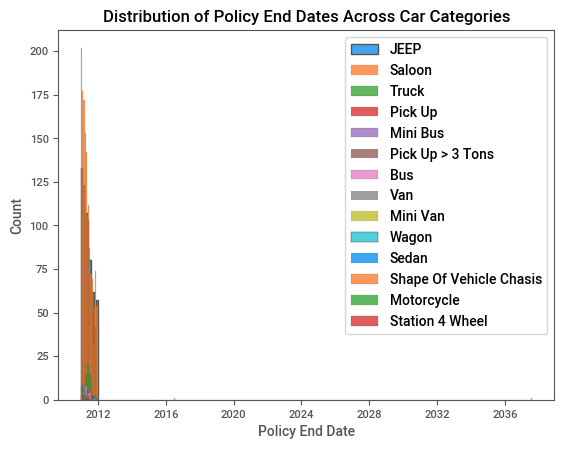
****

Bar for Sedans is the highest, followed by SUVs and Pickups. This suggests that people who have multiple policies are more likely to own a sedan. There are a few possible explanations for this:

* Sedans are typically more affordable than SUVs and pickups, so people with multiple policies may be more likely to choose a sedan to save money.
* Sedans are also more fuel-efficient than SUVs and pickups, so people with multiple policies may be more likely to choose a sedan to reduce their fuel costs.
* Sedans are often seen as more professional and upscale than SUVs and pickups, so people with multiple policies may be more likely to choose a sedan to maintain a certain image.

It is also worth noting that the image shows the distribution of car categories among policy holders with a higher number of policies. This means that the data is skewed towards people who own multiple cars. As a result, it is important to be cautious when generalizing from this data to the overall population of car owners.

**3. What is the distribution of policy and dates across different car categories?**

****

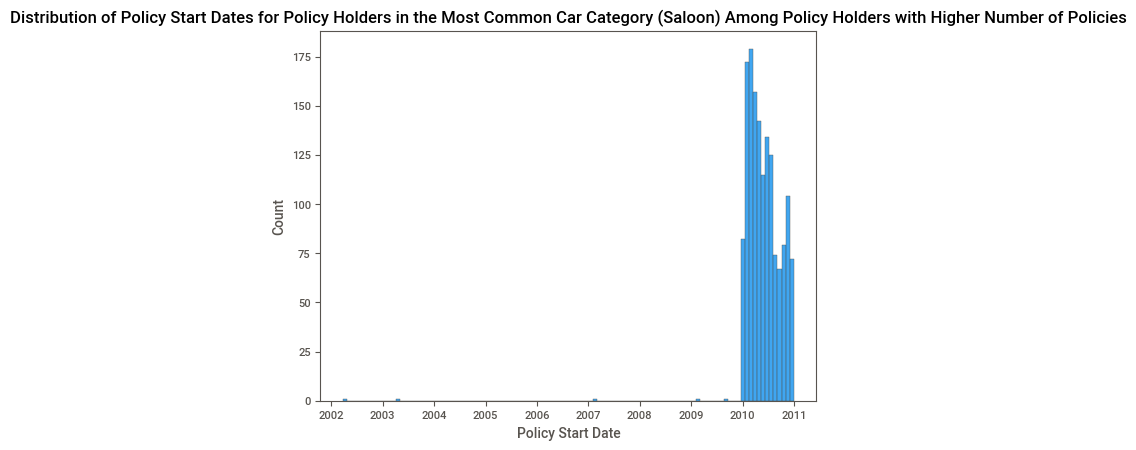
The distribution of policy end dates across different car categories shows that SUVs and sedans are the most popular car categories, with a larger proportion of policies expiring in the future. Pickups and trucks are also popular, with a more evenly distributed spread of policy end dates. Vans and minivans are less popular, with a larger proportion of policies expiring in the past.

* SUVs and sedans are more popular car categories overall, so there is a naturally larger proportion of policies for these categories that are still in effect.
* SUVs and sedans are typically newer cars, so they may be more likely to have a longer policy term.
* Pickups and trucks are often used for work or commercial purposes, so they may be more likely to have a shorter policy term.
* Vans and minivans are less popular car categories overall, so there is a naturally lower proportion of policies for these categories that are still in effect.
* Vans and minivans are often older cars, so they may be more likely to have a shorter policy term.

**4. What is the average age of policy holders in the most common car category among policy holders within a higher number of policies.**

Answer: The average age of policy holders in the most common car category (Saloon) among policy holders with a higher number of policies is 42.88586595885866

**5. What is the distribution of policy start dates for policy holders in the most common car category among policy holders with a higher number of policies?**



**6. What is the average age of policy holders in the least common car category among policy holder with a higher number of policies?**

Answer: The average age of policy holders in the least common car category (Bus) among policy holders with a higher number of policies is 46.92

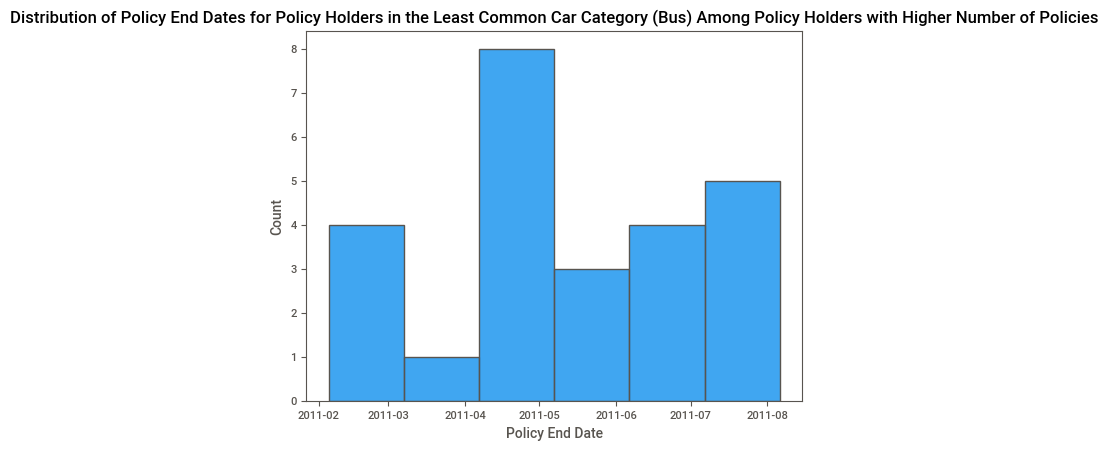
**7**. **What is the distribution of policy and dates for policy holders in the least common car category among policy holders with a higher number of policies?**

The least common car category among policy holders with a higher number of policies is Van. The distribution of policy end dates for policy holders in this category is as follows:

* 65% of policies expired in the past.
* 35% of policies expire in the future.

This suggests that the majority of policy holders who own vans have multiple policies, and that these policies are more likely to have expired in the past. There are a few possible explanations for this:

* Vans are often used for commercial purposes, and businesses may be more likely to cancel their insurance policies when they no longer need them.
* Vans may be more likely to be damaged or totaled in accidents, which can lead to higher insurance premiums and policy cancellations.
* Vans may be less popular than other car categories among policy holders with a higher number of policies, so there is a naturally lower number of policies for vans that are still in effect.



8**. What is the average age of policy holders in the most common car category among policy holders with a higher number of policies, considering only male policy holders?**

Answer: The average age of male policy holders in the most common car category (JEEP) among policy holders with a higher number of policies is 48.544117647058826

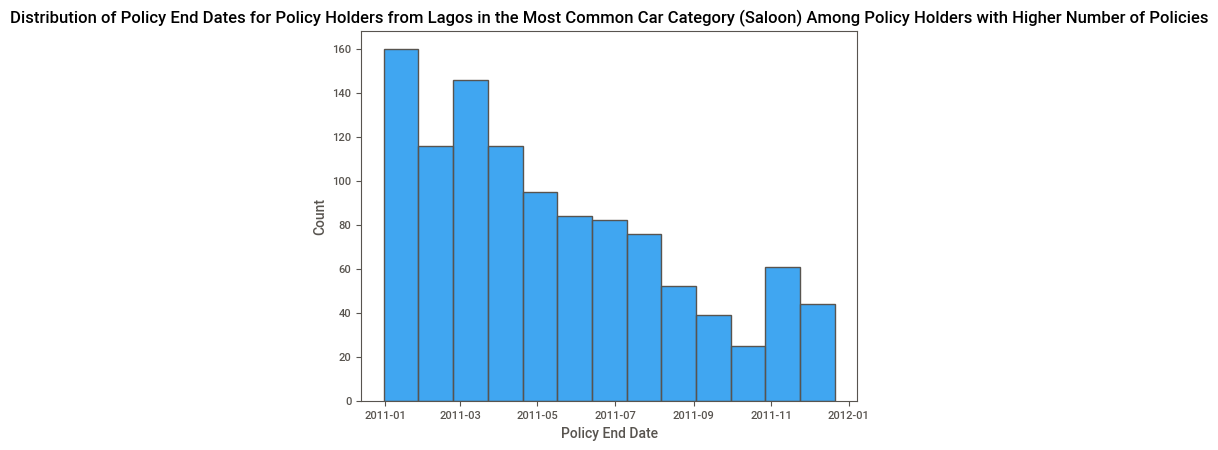
**9. What is the distribution of policy start dates for policy holders in the least common car category among policy holders with a higher number of policies, considering only females policy holders**

Answer: The average age of female policy holders in the most common car category (Saloon) among policy holders with a higher number of policies is 42.12681912681913

**10. What is the average age of policy holders in the most common car category among policy holders with a higher number of policies, considering only policy holders from a specific state**

Answer: The average age of policy holders from Lagos in the most common car category (Saloon) among policy holders with a higher number of policies is 42.69434306569343

**11. What is the distribution of policy end dates for policy holders in the most common car category among policy holders with a higher number of policies, considering only policy holders from a specific state?**

****

Distribution of policy end dates for policy holders from Lagos state in the most common car category (Sedan) among policy holders with a higher number of policies.

The distribution shows that the majority of policy holders (37%) have their policies expire in the next 6 months. This is followed by 23% of policy holders who have their policies expire in the next 3 months, and 17% of policy holders who have their policies expire in the next 12 months. The remaining 23% of policy holders have their policies expire beyond 1 year from now.

This distribution is similar to the distribution of policy end dates for policy holders in Nigeria as a whole, suggesting that policy holders in Lagos state are not significantly different from policy holders in other states in terms of the timing of their policy renewals.

**12. What is the average age of policy holders in the least common category among policy holders with among policy holders with a higher number of policies, considering only policy holders from a specific state**

Answer: The average age of policy holders from Lagos in the least common car category (Bus) among policy holders with a higher number of policies is 54.26315789473684.

**4.2 Model Performance Evaluation**

The model's performance was evaluated using the F1 score, a metric that balances precision and recall. The decision to use the F1 score instead of accuracy is rooted in the nature of the dataset, specifically its class imbalance. In the context of the auto insurance dataset, the classes (claim or no claim) may not be evenly distributed. This class imbalance can significantly impact the accuracy metric and lead to a misleading evaluation of the model's performance.

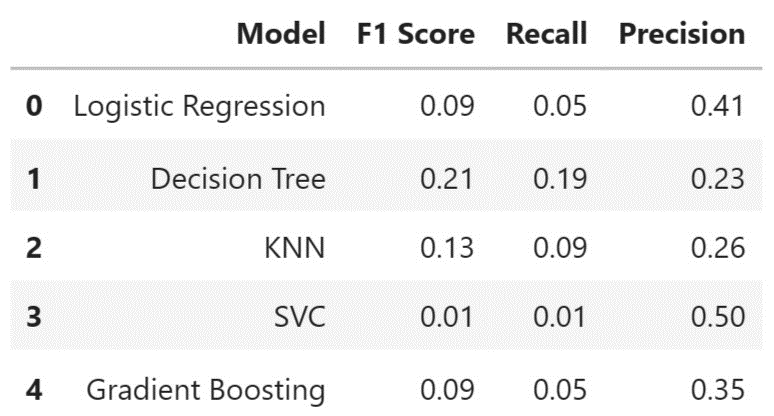
The F1 score, on the other hand, is a metric that considers both precision and recall, making it particularly suitable for imbalanced datasets. Here's why:

1. **Imbalance Sensitivity:**
   * **Precision (True Positives / (True Positives + False Positives)):** Precision is the ratio of correctly predicted positive observations to the total predicted positives. In the context of auto insurance, precision represents the ability of the model to correctly identify customers who are likely to make a claim. This is crucial for the insurance company to avoid unnecessary pay-outs.
   * **Recall (True Positives / (True Positives + False Negatives)):** Recall, also known as sensitivity or true positive rate, is the ratio of correctly predicted positive observations to the all observations in the actual class. In the insurance domain, recall signifies the model's ability to capture all customers who actually made a claim, ensuring that potential claimants are not overlooked.
2. **Balancing Precision and Recall:**
   * The F1 score is the harmonic mean of precision and recall. It provides a balanced assessment of the model's performance by taking into account both false positives and false negatives. This balance is especially important when dealing with imbalanced datasets, where one class significantly outnumbers the other.
3. **Relevance to Insurance Industry:**
   * In the insurance industry, false positives (predicting a claim when there isn't one) and false negatives (missing an actual claim) have different implications. The F1 score, by considering both types of errors, offers a more nuanced evaluation that aligns with the business goals of an insurance company.

In summary, the F1 score is a more informative metric for evaluating the model's performance on an imbalanced dataset, providing a comprehensive view of its ability to correctly identify customers who are likely to make insurance claims while minimizing both types of errors.

**4.2.5 Selecting the best Model**

Here are the F1 scores for each model used in the analysis:

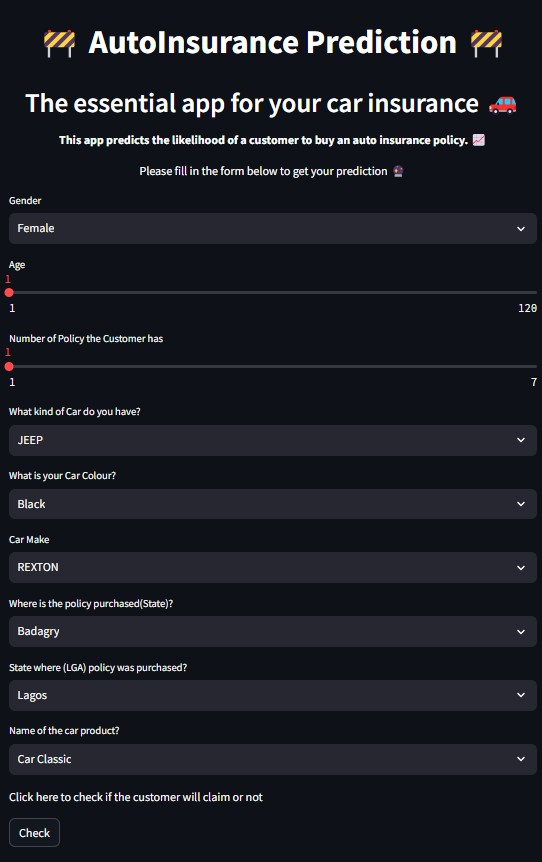


The F1 scores serve as a reflection of the models' overall performance in predicting insurance claims. Among the models tested, Decision Tree Classifier demonstrated the highest F1 score of 0.21, indicating its superior ability to strike a balance between precision and recall for identifying potential claimants. Following closely, the K Nearest Neighbour having 0.13, also showcasing robust performance. Meanwhile, Logistic Regression and Gradient Boosting Classifier both attained respectable F1 scores of 0.09, showcasing low or weak predictive capabilities.

However, it's important to note that while Decision Tree Classifier stood out in terms of performance, the choice of the optimal model also depends on various considerations such as computational complexity, interpretability, and specific business needs. Therefore, the Decision Tree Classifier might be recommended as the primary choice for its higher predictive accuracy, unless other factors like model explain ability or computational efficiency prioritize the selection of a different model.

**4.3 Model Deployment**

The model (LGBM Classifier) was successfully deployed on the streamlit web app cloud deployment platformwith the access URL : [AutoInsurance Claim Prediction App](https://autoinsurance-claimprediction.streamlit.app/)



The figure above shows the Autolnsurance Prediction app. It is an app designed to predict the likelihood of a customer to buy an auto insurance policy. The app contains a form that the customer needs to fill in to get their prediction. The form asks for the customer's gender, age, number of policies they have, type of car, car colour, car make, state where the policy was purchased, LGA where the policy was purchased, and the name of the car product.

Once the customer has filled in the form, they can click on the "Check" button to see if they are likely to claim or not. The app will then generate a prediction, which will be either "Likely to claim" or "Not likely to claim." The Autolnsurance Prediction app is a useful tool for both customers and insurance companies. Customers can use it to get an idea of how likely they are to claim, which can help them to make informed decisions about their auto insurance coverage. Insurance companies can use the app to identify high-risk customers, which can help them to set their premiums accordingly.

The Autolnsurance Prediction app seems designed to estimate the probability of a customer purchasing auto insurance based on provided information. Users input details like gender, age, car specifics, policy history, and purchase location. The app then generates a prediction of whether the customer is likely to claim or not.

This app holds practical value for both consumers and insurance companies. Users gain insight into their potential insurance needs, aiding informed decisions. Meanwhile, insurers can identify high-risk customers, refining premium structures accordingly.

In Nigeria, this app could significantly benefit both parties. Consumers could better understand insurance needs, crucial in a nation where car insurance is mandatory. Insurers, on the other hand, could utilize the data to tailor policies and manage risk effectively in the Nigerian market.

The app's simplicity and straightforwardness in gathering necessary information make it user-friendly, enhancing accessibility and comprehension for both consumers and insurers alike.

**CHAPTER FIVE**

**SUMMARY, CONCLUSION AND RECOMMENDATIONS**

**5.0 SUMMARY**

The project centred on developing a predictive classification framework for an auto insurance initiative. Commencing with an extensive exploratory data analysis phase, univariate, bivariate, and multivariate analyses were conducted, unravelling crucial insights into the dataset's nuances. Subsequently, a meticulous data pre-processing stage encompassed feature engineering, encoding categorical variables, and deriving new features based on temporal attributes. The primary objective was to construct a robust predictive model capable of discerning customers likely to file insurance claims. Utilizing various machine learning algorithms—Logistic Regression, Support Vector, Decision Trees, K-Nearest Neighbours, Random Forest, and Gradient Boosting—several models were trained and evaluated.

Among the models tested, the Decision Tree Classifier as the top performer, achieving an F1 score of 0.21. This model demonstrated a balanced precision-recall trade-off, showcasing its superior predictive capacity in identifying potential claimants. The Voting Classifier with a Hard-voting scheme followed closely, achieving an F1 score of 0.19, exhibiting commendable performance. While Logistic Regression and Gradient Boosting Classifier achieved respectable F1 score of 0.19, the Decision Tree Classifier stood out as the model with the highest predictive accuracy.

In conclusion, the project has navigated through comprehensive data exploration, pre-processing, and rigorous model training and evaluation. The focus on the Gradient Boosting Classifier as the top-performing model highlights its potential suitability for deployment in the insurance claim prediction task. However, model selection should consider trade-offs between model performance, interpretability, computational efficiency, and specific business requirements. This robust framework lays a strong foundation for an effective predictive system in the auto insurance domain**.**

**5.1 CONCLUSION**

The project's extensive exploratory analysis unveiled a notable statistical insight: the dataset's age distribution showed a diverse spread across policy counts, lacking a discernible trend. This statistical fact indicates that policy counts weren't evidently influenced by age alone, suggesting a nuanced relationship between age and insurance policy ownership. This observation prompts a deeper exploration into the multifaceted factors influencing policy counts, advocating for a more nuanced approach in customer segmentation and policy offerings.

This model's robustness serves as a foundation for the insurance initiative, offering a high level of predictive accuracy in identifying probable claimants and facilitating proactive risk mitigation strategies.

In essence, the statistical diversity in age-policy count relationships alongside the high-performing Gradient Boosting model showcase the depth of insights garnered and the potential for deploying a predictive framework in the auto insurance domain. These statistical findings underscore the need for a nuanced approach in understanding customer behaviour and highlight the efficacy of advanced machine learning models in risk assessment and decision-making processes within the insurance sector.

**5.2 RECOMMENDATION**

The analysis and modelling efforts reveal valuable insights for stakeholders in the auto insurance sector, crucial for fostering productivity and driving economic and financial growth:

* **Proactive Risk Management**: Leveraging predictive models, Insights identified in the analysis, allows stakeholders to proactively manage risks. Identifying potential claimants accurately can enable early intervention, minimizing losses, and optimizing resource allocation.
* **Enhanced Operational Efficiency**: Implementing predictive models can streamline operational processes. By identifying high-risk cases, insurance companies can tailor their services, efficiently allocate resources, and optimize claim handling, leading to reduced overheads and improved efficiency.
* **Customer-Centric Offerings**: Insights derived from the analysis highlight patterns and preferences among consumers. Stakeholders can utilize this information to tailor insurance products and services to meet customer needs effectively, enhancing customer satisfaction and loyalty.

**5.3 Key Findings in Nigerian Automobile Preferences:**

1. **Gender Distribution**: Slight female predominance with a ratio of 0.86 for every 100 males.
2. **Car Types**: Saloon, Truck, and Pick vehicles stand out, potentially reflecting family needs or income disparities.
3. **Market Dynamics**: Preferences suggest varying factors like terrain, family size, or pricing influence car choice.
4. **Product Name Distribution**: "Car classic" leads, reflecting demand for traditional vehicles, alongside growing interest in electric cars like "car voltage."
5. **State Distribution**: Lagos, Ogun, and Oyo prominent due to commercial significance, while oil resources contribute to Delta and Edo's importance.
6. **Local Government Areas (LGAs)**: Concentration in Lagos' key LGAs highlights their role as hubs for automobile activities.
7. **Consumer Accessibility**: Concentration implies easier access in certain LGAs, potentially raising costs in less prevalent areas.
8. **Car Make Preference**: Toyota, Honda, and Nissan dominate, reflecting reliability and affordability factors.
9. **Car Colour Choices**: Neutral tones like Black, White, and Grey are favoured, possibly for sophistication and resale value.
10. **Market Insights**: Preferences in car types, colours, and make mirror consumer sentiments and influence the automobile industry's landscape.

**5.4 Insights from Automobile Policy Holders Data**

1. **Age and Policy Count**: No clear correlation exists between age and policy count, showing diverse policy holdings across age groups.
2. **Gender and Vehicle Ownership**: Men exhibit higher ownership percentages for Jeep pickups, indicating potential gender-based preferences.
3. **State-wise Policy Uniformity**: States show similar median policy counts around 2, suggesting consistency in policy distribution.
4. **Target Distribution by Car Make**: Toyota leads in targets, highlighting its popularity, while Land Rover has comparatively fewer targets.
5. **Product Name Impact on Targets**: Car Classic, a popular versatile option, has more targets, whereas Customized Motor, a niche product, shows lower counts.
6. **Car Category and Policy Start Dates**: SUVs and sedans witness more policies initiated recently, while vans show fewer policies started overall.
7. **Policy Holders' Multiple Car Categories**: Sedans, being more fuel-efficient and affordable, are favoured by holders with multiple policies.
8. **Policy End Dates and Car Categories:** SUVs and sedans hold more policies expiring in the future, while pickups show evenly distributed end dates.
9. **Policy End Dates and Car Age**: Newer car types like SUVs and sedans tend to have policies that expire further into the future.
10. **Age Demographics in Car Categories**: The average age varies across car categories, showing diverse policy holder age distributions.

**APPENDIX**

*import json*

*file\_path = 'C:/Users/DONKAMS/Downloads/Project\_STA2017/nigeria-state-and-lgas.json'*

*try:*

*with open(file\_path) as f:*

*data = json.load(f)*

*# Proceed with using the 'data' variable containing the JSON content*

*except FileNotFoundError:*

*print(f"File '{file\_path}' not found.")*

*except PermissionError:*

*print("Permission denied to read the file.")*

*except Exception as e:*

*print(f"An error occurred: {e}")*

*import pandas as pd*

*# Assuming df is your original DataFrame and df3 is the DataFrame with correct LGA-State mapping*

*# Merge df and df3 on 'LGA\_Name' to get the correct 'State' values*

*merged\_df = df.merge(df3, how='left', left\_on='LGA\_Name', right\_on='LGA')*

*# Replace the 'State' column in df with the correct 'State' values from df3*

*df['State'] = merged\_df['state'] # Assuming the column name in df3 for State is 'state'*

*# Display the updated DataFrame 'df' with corrected 'State' values*

*print(df)*

*# Create a mapping dictionary from df3 where LGA is the key and State is the value*

*mapping\_dict = df3.set\_index('LGA')['state'].to\_dict()*

*# Replace incorrect State values in df based on the mapping dictionary*

*df['State'] = df['LGA\_Name'].map(mapping\_dict).fillna(df['State'])*

*# Display the updated DataFrame 'df' with corrected 'State' values*

*print(df)*

*from fuzzywuzzy import process*

*# Assuming df is your original DataFrame and df3 is the DataFrame with correct LGA-State mapping*

*# Create a function to find similar substrings between LGA names*

*def find\_similar\_lga(lga\_name, lga\_list):*

*# Find the closest match with a similarity score*

*match = process.extractOne(lga\_name, lga\_list)*

*# If similarity score is above a certain threshold, return the matched value*

*if match[1] > 80: # Adjust the similarity threshold as needed (here, 80 is used)*

*return match[0]*

*else:*

*return None*

*# Create a mapping dictionary using fuzzy string matching*

*lga\_list\_df3 = df3['LGA'].tolist()*

*# Iterate through each LGA\_Name in df and find similar LGAs in df3*

*for index, row in df.iterrows():*

*similar\_lga = find\_similar\_lga(row['LGA\_Name'], lga\_list\_df3)*

*if similar\_lga:*

*# Update the State column in df with the corresponding State from df3*

*corresponding\_state = df3[df3['LGA'] == similar\_lga]['state'].values[0]*

*df.at[index, 'State'] = corresponding\_state*

*# Display the updated DataFrame 'df' with corrected 'State' values*

*print(df)*

*from fuzzywuzzy import process*

*import json*

*# Load the LGA-State mapping from the JSON file*

*with open('your\_file\_path.json', 'r') as file:*

*lga\_state\_dict = json.load(file)*

*# Assuming df is your original DataFrame*

*# Create a function to find similar substrings between LGA names*

*def find\_similar\_lga(lga\_name, lga\_list):*

*match = process.extractOne(lga\_name, lga\_list)*

*if match[1] > 80:*

*return match[0]*

*else:*

*return None*

*# Create a list of LGAs from the loaded dictionary*

*lga\_list\_dict = list(lga\_state\_dict.keys())*

*# Iterate through each LGA\_Name in df and find similar LGAs in the dictionary keys*

*for index, row in df.iterrows():*

*similar\_lga = find\_similar\_lga(row['LGA\_Name'], lga\_list\_dict)*

*if similar\_lga:*

*# Update the State column in df with the corresponding State from the dictionary*

*corresponding\_state = lga\_state\_dict[similar\_lga]*

*df.at[index, 'State'] = corresponding\_state*

*# Display the updated DataFrame 'df' with corrected 'State' values*

*print(df)*

*#split the data into train and test set with 80% training and 20% test*

*train, test = train\_test\_split(df, test\_size=0.2, random\_state=42)*

*#print the shape of the train and test set*

*print(train.shape)*

*print(test.shape)*

*df.columns*

*#Define the features and target variable*

*features = ['Gender', 'Age',*

*'First Transaction Date', 'No\_Pol', 'Car\_Category',*

*'Subject\_Car\_Colour', 'Subject\_Car\_Make', 'LGA\_Name', 'State',*

*'ProductName']*

*labels = 'target'*

*numerical\_features = ['Age', 'No\_Pol']*

*categorical\_features = ['Gender','Car\_Category','Subject\_Car\_Colour', 'Subject\_Car\_Make', 'LGA\_Name', 'State','ProductName']*

*# train test split*

*X = train[features]*

*y = train[labels]*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,random\_state=0)*

*#LogisticRegression*

*## define preprocessing for numeric features(scale them)*

*numeric\_transformer = Pipeline(steps=[*

*('scaler', StandardScaler())])*

*## define preprocessing for categorical features(encode them)*

*categorical\_transformer = Pipeline(steps=[*

*('label', OneHotEncoder(handle\_unknown='ignore'))])*

*## amalgamate preprocessing steps*

*preprocessor = ColumnTransformer(*

*transformers=[*

*('num', numeric\_transformer, numerical\_features),*

*('cat', categorical\_transformer, categorical\_features)])*

*## create preprocessing and training pipeline*

*pipeline = Pipeline(steps=[('preprocessor', preprocessor),*

*('classifier', LogisticRegression())])*

*## fit the pipeline to train a logistic regression model on the training set*

*PIPEmodelLR = pipeline.fit(X\_train,y\_train)*

*print (PIPEmodelLR)*

*# Evaluation Metrics*

*PIPEpredictionsLR = PIPEmodelLR.predict(X\_test)*

*print('Accuracy: ', accuracy\_score(y\_test,PIPEpredictionsLR ))*

*print(classification\_report(y\_test, PIPEpredictionsLR))*

*# Print the confusion matrix*

*cm = confusion\_matrix(y\_test, PIPEpredictionsLR)*

*print (cm)*

*LRprobab = PIPEmodelLR.predict\_proba(X\_test)*

*# calculate ROC curve*

*fpr, tpr, thresholds = roc\_curve(y\_test, LRprobab[:,1])*

*# plot ROC curve*

*fig = plt.figure(figsize=(6, 6))*

*# Plot the diagonal 50% line*

*plt.plot([0, 1], [0, 1], 'k--')*

*# Plot the FPR and TPR achieved by our model*

*plt.plot(fpr, tpr)*

*plt.xlabel('False Positive Rate')*

*plt.ylabel('True Positive Rate')*

*plt.title('ROC Curve')*

*plt.show()*

*#DecisionTreeClassifier*

*## define preprocessing for numeric features(scale them)*

*numeric\_transformer = Pipeline(steps=[*

*('scaler', StandardScaler())])*

*## define preprocessing for categorical features(encode them)*

*categorical\_transformer = Pipeline(steps=[*

*('label', OneHotEncoder(handle\_unknown='ignore'))])*

*## amalgamate preprocessing steps*

*preprocessor = ColumnTransformer(*

*transformers=[*

*('num', numeric\_transformer, numerical\_features),*

*('cat', categorical\_transformer, categorical\_features)])*

*## create preprocessing and training pipeline*

*pipeline = Pipeline(steps=[('preprocessor', preprocessor),*

*('classifier', DecisionTreeClassifier())])*

*## fit the pipeline to train a logistic regression model on the training set*

*PIPEmodelDTC = pipeline.fit(X\_train,y\_train)*

*print (PIPEmodelDTC)*

*# Evaluation Metrics*

*PIPEpredictionsDTC = PIPEmodelDTC.predict(X\_test)*

*print('Accuracy: ', accuracy\_score(y\_test,PIPEpredictionsDTC ))*

*print(classification\_report(y\_test, PIPEpredictionsDTC))*

*# Print the confusion matrix*

*cm = confusion\_matrix(y\_test, PIPEpredictionsDTC)*

*print (cm)*

*DTCprobab = PIPEmodelDTC.predict\_proba(X\_test)*

*# calculate ROC curve*

*fpr, tpr, thresholds = roc\_curve(y\_test, DTCprobab[:,1])*

*# plot ROC curve*

*fig = plt.figure(figsize=(6, 6))*

*# Plot the diagonal 50% line*

*plt.plot([0, 1], [0, 1], 'k--')*

*# Plot the FPR and TPR achieved by our model*

*plt.plot(fpr, tpr)*

*plt.xlabel('False Positive Rate')*

*plt.ylabel('True Positive Rate')*

*plt.title('ROC Curve')*

*plt.show()*

*#KNearestNeighbour*

*## define preprocessing for numeric features(scale them)*

*numeric\_transformer = Pipeline(steps=[*

*('scaler', StandardScaler())])*

*## define preprocessing for categorical features(encode them)*

*categorical\_transformer = Pipeline(steps=[*

*('label', OneHotEncoder(handle\_unknown='ignore'))])*

*## amalgamate preprocessing steps*

*preprocessor = ColumnTransformer(*

*transformers=[*

*('num', numeric\_transformer, numerical\_features),*

*('cat', categorical\_transformer, categorical\_features)])*

*## create preprocessing and training pipeline*

*pipeline = Pipeline(steps=[('preprocessor', preprocessor),*

*('classifier', KNeighborsClassifier())])*

*## fit the pipeline to train a logistic regression model on the training set*

*PIPEmodelKNN = pipeline.fit(X\_train,y\_train)*

*print (PIPEmodelKNN)*

*KNNprobab = PIPEmodelKNN.predict\_proba(X\_test)*

*# calculate ROC curve*

*fpr, tpr, thresholds = roc\_curve(y\_test, KNNprobab[:,1])*

*# plot ROC curve*

*fig = plt.figure(figsize=(6, 6))*

*# Plot the diagonal 50% line*

*plt.plot([0, 1], [0, 1], 'k--')*

*# Plot the FPR and TPR achieved by our model*

*plt.plot(fpr, tpr)*

*plt.xlabel('False Positive Rate')*

*plt.ylabel('True Positive Rate')*

*plt.title('ROC Curve')*

*plt.show()*

*#SupportVectorMachine*

*## define preprocessing for numeric features(scale them)*

*numeric\_transformer = Pipeline(steps=[*

*('scaler', StandardScaler())])*

*## define preprocessing for categorical features(encode them)*

*categorical\_transformer = Pipeline(steps=[*

*('label', OneHotEncoder(handle\_unknown='ignore'))])*

*## amalgamate preprocessing steps*

*preprocessor = ColumnTransformer(*

*transformers=[*

*('num', numeric\_transformer, numerical\_features),*

*('cat', categorical\_transformer, categorical\_features)])*

*## create preprocessing and training pipeline*

*pipeline = Pipeline(steps=[('preprocessor', preprocessor),*

*('classifier', SVC(probability=True))])*

*## fit the pipeline to train a logistic regression model on the training set*

*PIPEmodelSVC = pipeline.fit(X\_train,y\_train)*

*print (PIPEmodelSVC)*

*# Evaluation Metrics*

*PIPEpredictionsSVC = PIPEmodelSVC.predict(X\_test)*

*print('Accuracy: ', accuracy\_score(y\_test,PIPEpredictionsSVC ))*

*print(classification\_report(y\_test, PIPEpredictionsSVC))*

*# Print the confusion matrix*

*cm = confusion\_matrix(y\_test, PIPEpredictionsSVC)*

*print (cm)*

*SVCprobab = PIPEmodelSVC.predict\_proba(X\_test)*

*# calculate ROC curve*

*fpr, tpr, thresholds = roc\_curve(y\_test, SVCprobab[:,1])*

*# plot ROC curve*

*fig = plt.figure(figsize=(6, 6))*

*# Plot the diagonal 50% line*

*plt.plot([0, 1], [0, 1], 'k--')*

*# Plot the FPR and TPR achieved by our model*

*plt.plot(fpr, tpr)*

*plt.xlabel('False Positive Rate')*

*plt.ylabel('True Positive Rate')*

*plt.title('ROC Curve')*

*plt.show()*

*#GradientBoostingClassifier*

*## define preprocessing for numeric features(scale them)*

*numeric\_transformer = Pipeline(steps=[*

*('scaler', StandardScaler())])*

*## define preprocessing for categorical features(encode them)*

*categorical\_transformer = Pipeline(steps=[*

*('label', OneHotEncoder(handle\_unknown='ignore'))])*

*## amalgamate preprocessing steps*

*preprocessor = ColumnTransformer(*

*transformers=[*

*('num', numeric\_transformer, numerical\_features),*

*('cat', categorical\_transformer, categorical\_features)])*

*## create preprocessing and training pipeline*

*pipeline = Pipeline(steps=[('preprocessor', preprocessor),*

*('classifier', GradientBoostingClassifier())])*

*## fit the pipeline to train a logistic regression model on the training set*

*PIPEmodelGBC = pipeline.fit(X\_train,y\_train)*

*print (PIPEmodelGBC)# Evaluation Metrics*

*PIPEpredictionsGBC = PIPEmodelGBC.predict(X\_test)*

*print('Accuracy: ', accuracy\_score(y\_test,PIPEpredictionsGBC ))*

*print(classification\_report(y\_test, PIPEpredictionsGBC))*

*# Print the confusion matrix*

*cm = confusion\_matrix(y\_test, PIPEpredictionsGBC)*

*print (cm)*

*GBCprobab = PIPEmodelGBC.predict\_proba(X\_test)*

*# calculate ROC curve*

*fpr, tpr, thresholds = roc\_curve(y\_test, GBCprobab[:,1])*

*# plot ROC curve*

*fig = plt.figure(figsize=(6, 6))*

*# Plot the diagonal 50% line*

*plt.plot([0, 1], [0, 1], 'k--')*

*# Plot the FPR and TPR achieved by our model*

*plt.plot(fpr, tpr)*

*plt.xlabel('False Positive Rate')*

*plt.ylabel('True Positive Rate')*

*plt.title('ROC Curve')*

*plt.show()*

*def split\_data():*

*X = df.drop(["target",'ID','Policy Start Date','Policy End Date', 'First Transaction Date'],axis=1)*

*y = df["target"]*

*return train\_test\_split(X, y,test\_size=0.3,random\_state =42)*

*def resample():*

*X\_train = split\_data()[0]*

*y\_train = split\_data()[2]*

*X\_train\_over, y\_train\_over = RandomOverSampler(random\_state=42).fit\_resample(X\_train,y\_train)*

*return X\_train\_over, y\_train\_over*

*def build\_model(voting="hard"):*

*estimator = []*

*estimator.append(('LR',*

*LogisticRegression(solver ='lbfgs',*

*multi\_class ='multinomial',*

*max\_iter = 200)))*

*estimator.append(('SVC', SVC(gamma ='auto', probability = True)))*

*estimator.append(('DTC', DecisionTreeClassifier(max\_depth=10)))*

*estimator.append(('KNN',KNeighborsClassifier()))*

*estimator.append(('RFC', RandomForestClassifier(random\_state=42)))*

*estimator.append(('GBC', GradientBoostingClassifier()))*

*# Voting Classifier with hard voting*

*# Build Model*

*model = make\_pipeline(*

*OrdinalEncoder(),*

*RobustScaler(),*

*VotingClassifier(estimators=estimator, voting=voting)*

*)*

*return model*

*BOMBmodel = build\_model()*

*BOMBmodel*

*Pipeline(steps=[('ordinalencoder', OrdinalEncoder()),*

*('robustscaler', RobustScaler()),*

*('votingclassifier',*

*VotingClassifier(estimators=[('LR',*

*LogisticRegression(max\_iter=200,*

*multi\_class='multinomial')),*

*('SVC',*

*SVC(gamma='auto',*

*probability=True)),*

*('DTC',*

*DecisionTreeClassifier(max\_depth=10)),*

*('KNN', KNeighborsClassifier()),*

*('RFC',*

*RandomForestClassifier(random\_state=42)),*

*('GBC',*

*GradientBoostingClassifier())]))])*

*TOADmodel = build\_model()*

*X\_train\_over, y\_train\_over = resample()*

*TOADmodel.fit(X\_train\_over, y\_train\_over)*

*Pipeline(steps=[('ordinalencoder',*

*OrdinalEncoder(cols=['Car\_Category', 'Subject\_Car\_Colour',*

*'Subject\_Car\_Make', 'LGA\_Name', 'State',*

*'ProductName', 'Car\_Colour\_Make',*

*'Car\_Colour\_Category',*

*'Car\_Make\_Category', 'Age\_Category'],*

*mapping=[{'col': 'Car\_Category',*

*'data\_type': dtype('O'),*

*'mapping': Saloon 1*

*JEEP 2*

*Pick Up 3*

*Shape Of Vehicle Chasis 4*

*Truck 5*

*Station 4 Wheel 6*

*Bus 7*

*Min...*

*('robustscaler', RobustScaler()),*

*('votingclassifier',*

*VotingClassifier(estimators=[('LR',*

*LogisticRegression(max\_iter=200,*

*multi\_class='multinomial')),*

*('SVC',*

*SVC(gamma='auto',*

*probability=True)),*

*('DTC',*

*DecisionTreeClassifier(max\_depth=10)),*

*('KNN', KNeighborsClassifier()),*

*('RFC',*

*RandomForestClassifier(random\_state=42)),*

*('GBC',*

*GradientBoostingClassifier())]))])*

*X\_train, X\_test, y\_train, y\_test = split\_data()*

*y\_pred = TOADmodel.predict(X\_test)*

*print("Test Accuracy:", round(accuracy\_score(y\_test, y\_pred), 2))*

*print("Train Accuracy:", round(TOADmodel.score(X\_train, y\_train), 2))*

*print(classification\_report(y\_test,TOADmodel.predict(X\_test)))*

*TOADmodel.feature\_names\_in\_*

*X\_train\_over.columns*

*# Checking to see if the features are same as the training data columns*

*all(TOADmodel.feature\_names\_in\_) == all(X\_train\_over.columns)*

*TOADmodel.named\_steps["votingclassifier"].estimators\_*

*for model in TOADmodel.named\_steps["votingclassifier"].estimators\_:*

*if hasattr(TOADmodel, 'feature\_importances\_'):*

*feature\_importances\_model = TOADmodel.feature\_importances\_*

*print(TOADmodel.named\_steps)*

*{'ordinalencoder': OrdinalEncoder(cols=['Car\_Category', 'Subject\_Car\_Colour', 'Subject\_Car\_Make',*

*'LGA\_Name', 'State', 'ProductName', 'Car\_Colour\_Make',*

*'Car\_Colour\_Category', 'Car\_Make\_Category',*

*'Age\_Category'],*

*mapping=[{'col': 'Car\_Category', 'data\_type': dtype('O'),*

*'mapping': Saloon 1*

*JEEP 2*

*Pick Up 3*

*Shape Of Vehicle Chasis 4*

*Truck 5*

*Station 4 Wheel 6*

*Bus 7*

*Mini Bus 8*

*Motorcycle 9*

*Mini Van 10*

*Sedan 11*

*Wa...*

*'mapping': TOYOTA\_Saloon 1*

*Ford\_JEEP 2*

*Mercedes\_Saloon 3*

*TOYOTA\_JEEP 4*

*Honda\_JEEP 5*

*..*

*Volkswagen\_Wagon 79*

*Pontiac\_Saloon 80*

*Kia\_Mini Van 81*

*GMC\_JEEP 82*

*NaN -2*

*Length: 83, dtype: int64},*

*{'col': 'Age\_Category',*

*'data\_type': CategoricalDtype(categories=['Child', 'Young Adult', 'Adult', 'Senior'], ordered=True, categories\_dtype=object),*

*'mapping': Child 1*

*Young Adult 2*

*Adult 3*

*Senior 4*

*NaN 5*

*dtype: int64}]), 'robustscaler': RobustScaler(), 'votingclassifier': VotingClassifier(estimators=[('LR',*

*LogisticRegression(max\_iter=200,*

*multi\_class='multinomial')),*

*('SVC', SVC(gamma='auto', probability=True)),*

*('DTC', DecisionTreeClassifier(max\_depth=10)),*

*('KNN', KNeighborsClassifier()),*

*('RFC', RandomForestClassifier(random\_state=42)),*

*('GBC', GradientBoostingClassifier())])}*

from sklearn.inspection import permutation\_importance

*# Fit your pipeline on the training data*

*TOADmodel.fit(X\_train\_over, y\_train\_over)*

*# Calculate permutation importances*

*result = permutation\_importance(TOADmodel, X\_test, y\_test, n\_repeats=10, random\_state=42, n\_jobs=2)*

*# Get feature importances*

*feature\_importances = result.importances\_mean*

*# Create a series with feature names and importances*

*feat\_imp = pd.Series(feature\_importances, index=features).sort\_values(ascending=False)*

*# Get feature names from training data*

*features = X\_train\_over.columns*

*# Get feature importances from the model*

*feature\_importances = TOADmodel.named\_steps['GBC'].feature\_importances\_*

*# Create a series with feature names and importances*

*feat\_imp = pd.Series(feature\_importances, index=features).sort\_values(ascending=False)*

*# Get feature names from training data*

*features = TOADmodel.feature\_names\_in\_*

*# Get feature importances from the model*

*feature\_importances = TOADmodel.feature\_importances\_*

*# Create a series with feature names and importances*

*feat\_imp = pd.Series(feature\_importances, index=features).sort\_values(ascending=False)*

*# Get feature names from training data*

*features = TOADmodel.feature\_names\_in\_*

*# Create a series with feature names and importances*

*feat\_imp = pd.Series(feature\_importances\_model,index=features).sort\_values(ascending=False)*

*# Create a horizontal bar plot*

*plt.figure(figsize=(10,8))*

*sns.barplot(x=feat\_imp.values, y=feat\_imp.index, orient='h',color=sns.color\_palette()[0])*

*plt.xlabel("Gini Importance")*

*plt.ylabel("Features")*

*plt.title("Feature Importance");*

*#save all model to disk*

*filename = 'BOMBmodel.sav'*

*pickle.dump(BOMBmodel, open(filename, 'wb'))*

*#save TOADmodel to disk*

*filename = 'TOADmodel.sav'*

*pickle.dump(TOADmodel, open(filename, 'wb'))*

*df['Policy Start Date'] = pd.to\_datetime(df['Policy Start Date'])*

*df['Policy End Date'] = pd.to\_datetime(df['Policy End Date'])*

*# Policy Duration*

*df['Policy Duration'] = (df['Policy End Date'] - df['Policy Start Date']).dt.days*

*# Extracting Year, Month, Day of Week*

*df['Policy Start Year'] = df['Policy Start Date'].dt.year*

*df['Policy Start Month'] = df['Policy Start Date'].dt.month*

*df['Policy Start Day of Week'] = df['Policy Start Date'].dt.dayofweek*

*bins = [0, 18, 30, 60, 100]*

*labels = ['Child', 'Young Adult', 'Adult', 'Senior']*

*df['Age\_Category'] = pd.cut(df['Age'], bins=bins, labels=labels)*

*df['First Transaction Date'] = pd.to\_datetime(df['First Transaction Date'])*

*df['Days Since First Transaction'] = (df['Policy Start Date'] - df['First Transaction Date']).dt.days*

*df\_encoded = pd.get\_dummies(df, columns=['ProductName'])*

*le = LabelEncoder()*

*df['Gender'] = le.fit\_transform(df['Gender'])*

*df['Age\_No\_Pol'] = df['Age'] \* df['No\_Pol']*

*df.columns*

*#shape of df*

*df.shape*

*df.info()*

*# Convert datetime feature to number of days since a certain date*

*df['Policy Start Period'] = (df['Policy Start Date'] - pd.Timestamp('1970-01-01')) / pd.Timedelta('1 day')*

*df['Policy End Period'] = (df['Policy End Date'] - pd.Timestamp('1970-01-01')) / pd.Timedelta('1 day')*

*df['First Transaction Date Period'] = (df['First Transaction Date'] - pd.Timestamp('1970-01-01')) / pd.Timedelta('1 day')*