MGT7216: Data Mining

Title : Predictive Analytics for Customer Purchases: Insights and Recommendations for Imperials Ltd

Name: Dhanush Mathighatta Shobhan Babu
Student ID: 40412492

Word count :2198

Table Of Content

Table Of Figures

1. Introduction and background

In the dynamic and ever-changing insurance sector, organisations are progressively utilising data analytics to improve their decision-making processes and customise their marketing tactics to address the requirements of potential clients more effectively. Imperials Ltd, a prominent participant in the insurance industry, is leading the way in this innovation by aiming to employ sophisticated analytics to enhance the sales of their life insurance offerings.

This report conducts a thorough analysis with the goal of forecasting which consumers are inclined to buy life insurance. It utilises a sample of historical data from the company's enormous customer database. Predictive analytics plays a crucial role in the insurance industry. Insurance firms can gain insights into client behaviour, risk assessment, and product optimisation by examining historical data. This strategic approach not only increases customer pleasure through the provision of customised products and services, but also greatly optimises operational efficiency and profitability (Hand, 2001).

This Report seeks to enhance the existing foundation by utilising descriptive and predictive analytics to discover prospective life insurance clients among Imperials Ltd's current customer base. This study aims to construct a model that accurately predicts life insurance purchase intent by conducting a thorough analysis of the dataset provided. The dataset includes variables such as Gender, Highest Level of Education, House Value, Age, Online Purchase Behaviour, Marital Status, Parenthood, Occupation, Mortgage Bracket, Homeownership, Regional Location, and Family Income Grade.

2. Literature review

Title of the paper	Year of Publication	Author	Conclusion
Life Insurance Prediction and Its Sustainability Using Machine Learning Approach	2023	Shamsuddin, S.N., Ismail, N. and Nur-Firyal, R	This study delves into machine learning techniques for predicting life insurance policyholders, with a specific emphasis on effectively managing imbalanced datasets. The findings highlight the effectiveness of the decision tree and Naïve Bayes models, particularly in terms of balanced accuracy, F1 score, and Geometric Mean (GM) for imbalanced data. The Logistic Regression (LR) models, particularly when combined with SMOTE sampling, exhibit a remarkable balanced accuracy of up to 67.02%, which signifies their robust predictive capability. This research highlights the crucial significance of choosing the right models and sampling methods to enhance predictions in life insurance. It provides valuable insights for improving the processes of identifying policyholders within the industry.
Risk prediction in life insurance industry using supervised learning algorithms	2018	Boodhun, N. and Jayabalan, M.	The focus of the literature study is to enhance risk assessment methodologies in the field of life insurance through the use of machine learning algorithms, including Multiple Linear Regression, Artificial Neural Networks, REPTree, and Random Tree technologies. In order to enhance the precision of the model, dimensionality reduction methods such as Correlation-Based Feature Selection (CFS) and Principal Components Analysis (PCA) are included. When evaluating several algorithms, it is evident that REPTree exhibits

			significant effectiveness, as it achieves the lowest Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) when combined with CFS. The aforementioned result highlights the enhanced predicting powers of the REPTree algorithm in the realm of life insurance risk evaluation.
Customer profitability forecasting using Big Data analytics: A case study of the insurance industry	2016	Fang, K., Jiang, Y. and Song, M.	This study aims to assess the profitability of insurance customers using Random Forest Regression (RFR) through the application of Big Data analytics. The study introduces a novel element by integrating liability reserve funds into the calculation of insurance customer profitability. This approach seeks to offer a more comprehensive representation of consumers' financial contributions. This comprehensive strategy considers both past and future purchase patterns in addition to cash flows. The results of the study suggest that Random Forest Regression exhibits superior performance compared to other models such as linear regression, decision trees, Support Vector Machines (SVM), and generalised boosted models when it comes to predicting client profitability. Moreover, the study clarifies the substantial impact of factors such as geographic location, age, insurance coverage, gender, and consumer origin on the profitability of insurance clients. In general, the review highlights the effectiveness of Random Forest Regression in utilising Big Data analytics for predictive modelling in the insurance industry.

An Ensemble Random Forest Algorithm for Insurance Big Data Analysis	2017	Weiwei Lin; Ziming Wu; Longxin Lin; Angzhan Wen; Jin Li	The ensemble random forest technique, optimized with Spark, outperforms traditional models like SVM and Logistic Regression in handling large, imbalanced insurance datasets. Using China Life Insurance Company's dataset, it achieves a 30.9% recall rate in identifying potential customers, indicating superior efficacy for imbalanced classification tasks. Out of 60,831 potential clients identified, approximately 14,600 (24%) made purchases. This contrasts with a 4% recall rate from the old method. These findings underscore the method's effectiveness in forecasting potential consumers in skewed datasets, suggesting advancements in insurance client targeting and acquisition strategies.
Improving insurers' loss reserve error prediction: Adopting combined unsupervised-supervised machine learning techniques in risk management	2022	Wookjae Heo In Jung Song	The study introduces a methodology merging unsupervised and supervised machine learning techniques to enhance prediction accuracy for loss reserve errors in insurers. Combining cluster analysis with algorithms like ANN, SVM, Gradient Boosting, and Adaptive Boosting, it shows Adaptive Boosting outperforms OLS in RMSE values (1.06, 1.12, 1.26, 1.32 for Clusters A, B, C, and D). This indicates notable improvement in prediction accuracy, particularly within certain clusters (A and B), suggesting insurer-specific attributes significantly impact the predictive performance of the machine learning model.

Supervised Machine Learning Algorithms: Classification and Comparison	2017	Osisanwo F.Y., Akinsola J.E.T., Awodele O., Hinmikaiye J. O., Olakanmi O., Akinjobi J.	Osisanwo et al. conduct an analysis of seven algorithms using a diabetes dataset in their paper titled "Supervised Machine Learning Algorithms: Classification and Comparison." The most accurate model is Support Vector Machine (SVM), followed by Naïve Bayes and Random Forest. Support Vector Machines (SVM) demonstrate the best accuracy rate of correctly classifying data (77.34%) and the lowest rate of incorrectly classifying data (22.66%), notably in the context of predicting positive diabetic outcomes. The study proposes that the selection of algorithms should be based on the attributes of the dataset and the number of cases in order to improve the accuracy of forecasts.
Predictive Data Modeling: Educational Data Classification and Comparative Analysis of Classifiers Using Python	2018	Pratiyush Guleria, Manu Sood	In this work, a range of classifiers were employed to classify student performance, including Decision Trees, K-Nearest Neighbours (KNN), Linear Discriminant Analysis (LDA), Gaussian Naive Bayes (NB), Logistic Regression (LR), and Support Vector Machines (SVM). The SVM, CART (Decision Tree), and KNN algorithms demonstrated higher levels of accuracy when applied to the categorization of student data across several subjects, suggesting their effectiveness in the field of educational data classification. Significantly, the Support Vector Machine (SVM) model exhibited remarkable accuracy, solidifying its position as the favoured option for predictive analytics and the classification of student performance.

Flash-flood susceptibility mapping based on XGBoost, random forest and boosted regression trees	2021	Rahebeh Abedi, Romulus Costache, Hossein Shafizadeh-Moghadam & Quoc Bao Pham	The research paper titled "Educational Data Classification and Comparative Analysis of Classifiers Using Python: A Predictive Data Modelling Study" examines the influence of machine learning classifiers on the field of education. Python assesses various machine learning algorithms, including Decision Trees, Neural Networks, Nearest Neighbour, Naive Bayes, SVM, CART, and KNN, for the purpose of identifying student data. The classification algorithms SVM, CART, and KNN are notable in the field of subject-wide classification. Among them, SVM is particularly well-suited for predictive analytics due to its high precision.
Decision Tree-Based Classification for Planetary Gearboxes' Condition Monitoring with the Use of Vibration Data in Multidimensional Symptom Space	2020	Lipinski, P., Brzychczy, E. and Zimroz, R.	By employing decision tree approaches, the study successfully attains a diagnostic accuracy of approximately 99% in monitoring the state of planetary gearboxes. This study evaluates the effectiveness of different decision tree algorithms, including classification and regression trees, by comparing them to alternative classifiers such as K-nearest neighbours, random forest, and AdaBoost. The evaluation is conducted using metrics such as the Gini index and entropy. The exceptional degree of precision demonstrated highlights the resilience of decision trees in effectively categorising gearbox conditions, particularly in dynamic operational situations.

A Data Mining & Knowledge Discovery Process Model	2009	Óscar Marbán, Gonzalo Mariscal and Javier Segovia	The scholarly report explores the models of data mining and knowledge discovery processes, emphasising the development and necessity of complete approaches in this field. The text provides a thorough evaluation of CRISP-DM, the widely used standard, highlighting shortcomings in its approach to project management, organisational, and quality-related elements. It suggests implementing a more organised engineering process model by incorporating neglected tasks to improve efficiency and organisation in data mining projects. This proposed approach emphasises the significance of comprehensive design and integration of additional processes to address the complexities of modern data mining difficulties, drawing upon ideas derived from software engineering standards.

An Assessment on Classification in Python Using Data Science	2021	Margaret Mary T, Soumya K, Ramanathan G, Clinton G	This paper evaluates classification algorithms in Python within the realm of Data Science, stressing the amalgamation of diverse models like decision trees, logistic regression, naive Bayes, support vector machines, random forests, and neural networks. Emphasizing Python's proficiency for data modeling and analysis via interfaces like PyCharm and Spyder, it positions Python as a central instrument for data science endeavors, elucidating insights derived from extensive data analysis including activation values and classification rules. The review culminates with practical applications of these methodologies in real-world contexts, while proposing avenues for further enhancement of classification techniques in Data Science utilizing Python.
Data mining to predict and prevent errors in health insurance claims processing	2010	Mohit Kumar, Rayid Ghani, Zhu-Song Mei. (2010)	This study introduces a machine learning-based system that tries to predict and address problems in the processing of health insurance claims. The primary objective is to reduce administrative expenses and minimise the need for claim rework. By analysing claims data from a big US health insurer, the technology demonstrates much improved accuracy in identifying errors compared to traditional methods, which could result in significant annual cost reductions. The system utilises feature selection, concept drift adaption, and active learning methods, along with auditor input, to continuously improve its performance. This showcases a scalable and feasible solution that can be used to other health insurance companies.

Feature selection based on artificial bee colony and gradient boosting decision tree	2019	Haidi Rao, Xianzhang Shi, Ahoussou Kouassi Rodrigue, Juanjuan Feng, Yingchun Xia, Mohamed Elhoseny, Xiaohui Yuan, Lichuan Gu. (2019)	This study introduces a novel approach to feature selection that combines Artificial Bee Colony (ABC) optimisation with Gradient Boosting Decision Tree (GBDT). The objective is to enhance the efficiency of handling high-dimensional data while also ensuring that the selected features are useful. The method demonstrates significant decrease in features, reaching up to 96.6%, and exhibits enhanced accuracy in classification across diverse datasets, including those related to breast cancer. The approach is notable for its ability to reduce feature redundancy and improve the quality of decision tree inputs. This demonstrates the efficacy of combining ABC and GBDT for advanced feature selection in the field of data analysis.

Comparison of the performance of GaussianNB Algorithm, the K Neighbors Classifier Algorithm, the Logistic Regression Algorithm, the Linear Discriminant Analysis Algorithm, and the Decision Tree Classifier Algorithm on same dataset	2022	Nisha Sawant, Dnyandev Ravindra, Khadapkar T	This study evaluates the efficacy of different classification algorithms, including as GaussianNB, K Neighbours Classifier, Logistic Regression, Linear Discriminant Analysis, and Decision Tree Classifier, in forecasting the influence of the Covid-19 epidemic on student academic achievement. The objective of this study is to determine the best accurate algorithm by utilising a dataset that is divided into 80% for training and 20% for testing. The RandomForestClassifier is also evaluated in terms of its applicability to various sorts of data. The findings of the investigation indicate that, with the exception of Linear Discriminant investigation and GaussianNB, all other algorithms demonstrated a prediction accuracy of 100%. This suggests that a considerable proportion of pupils exhibited enhanced performance amongst the epidemic.

Performance Analysis of SVC	2007	Mathias Wien, Heiko Schwarz, and Tobias Oelbaum	The literature review in the attached document focuses on evaluating the performance and enhancing encoder control and bitstream extraction for scalable video transmission in the Scalable Video Coding (SVC) version of H.264/AVC. The text outlines the main features of Support Vector Machines (SVC), focusing on the difficulties in controlling encoders and providing optimised settings specifically designed for scaling situations. Furthermore, the document showcases rate-distortion results for different SVC configurations, demonstrating the capacity of SVC to closely mimic the efficiency of single-layer H.264/AVC encoding while outperforming previous video coding standards such as MPEG-4 ASP. This paper highlights the potential of Support Vector Machines (SVC) in producing efficient and scalable video compression while minimising performance loss. It emphasises the importance of SVC in applications that require video transmission under various network conditions and device capabilities.

Fraud Detection in Automobile Insurance using a Data Mining Based Approach	2018	Ali Ghorbani and Sara Farzai	This research investigates the detection of fraudulent activities in the vehicle insurance industry through the utilisation of a data mining approach. The study specifically concentrates on the application of K-Means clustering to identify patterns that may indicate fraudulent claims. This study investigates a sample of 100 fraudulent instances obtained from many insurance companies in Iran. The research aims to shed light on prevalent fraud attributes, including anomalies in driver's licences and the timing of accidents. The accuracy of the procedure in spotting probable cases of fraud is underscored by its validation through expert discussions and real-world statistics. The study highlights the potential of data mining techniques in enhancing the fraud detection capabilities of the insurance business. It also suggests the need for additional research on fuzzy techniques and the impact of regional and market dynamics on fraud incidents.

3.0 Methodology

Before building any machine learning model, a number of basic tasks must be carried out. Data preparation involves preparing and cleaning the data before it can be used for analysis. Feature engineering refers to creating new features or transforming existing ones to improve the performance of the model. Model creation involves building the first model using the prepared data. Model improvement refers to improving the model's performance through techniques like as hyperparameter tuning or ensemble methods. Finally, model deployment is the process of making the produced model. Wirth, Rudiger, and Jochen Hipp developed a systematic framework known as Cross Industry Standard Process for Data Mining (CRISP-DM) to delineate the typical procedures involved in this process (Hotz, 2023). The CRISP-DM technique offers a well-defined and organised framework for addressing business problems. We will utilise the CRISP-DM methodology to tackle our business challenge. From this juncture, we have already completed the Business Understanding procedure. Next, we will proceed to address each of the remaining sections.

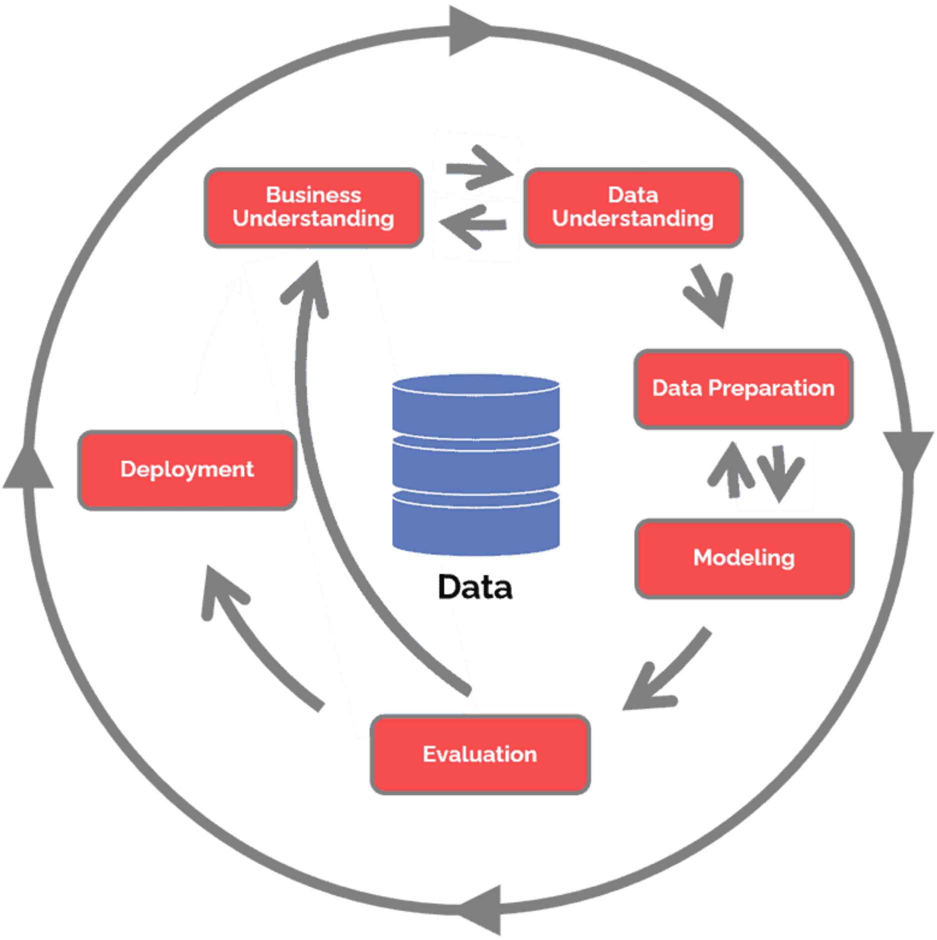
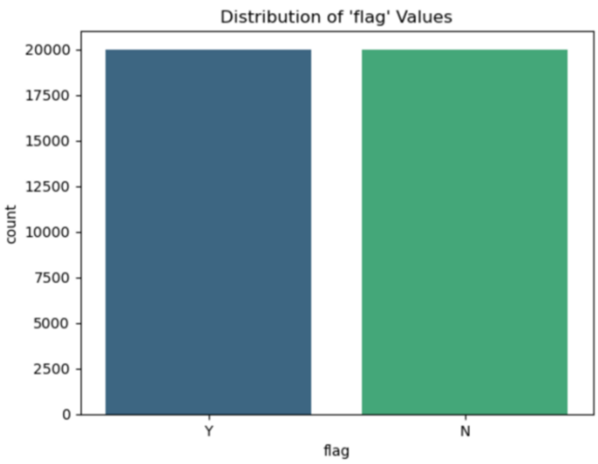


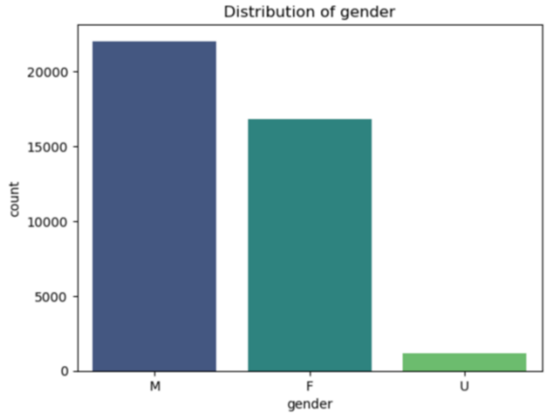
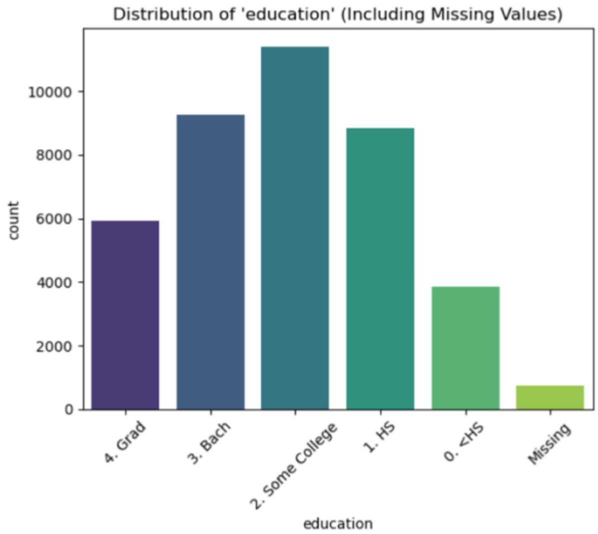
Fig 3.0: CRISP-DM Steps

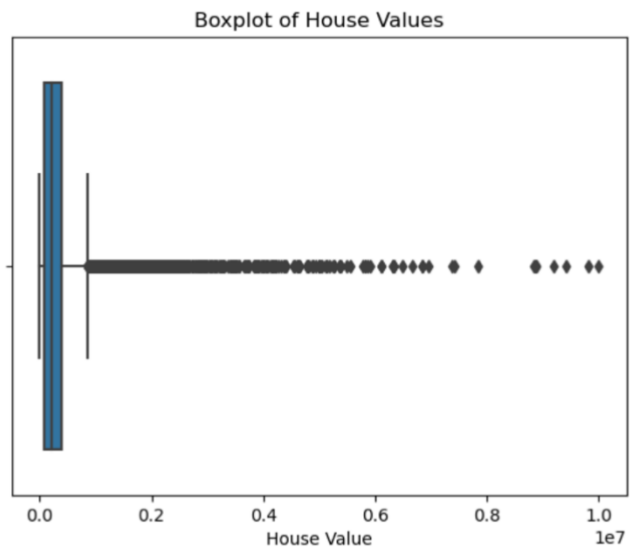
3.1 Data Pre-Processing

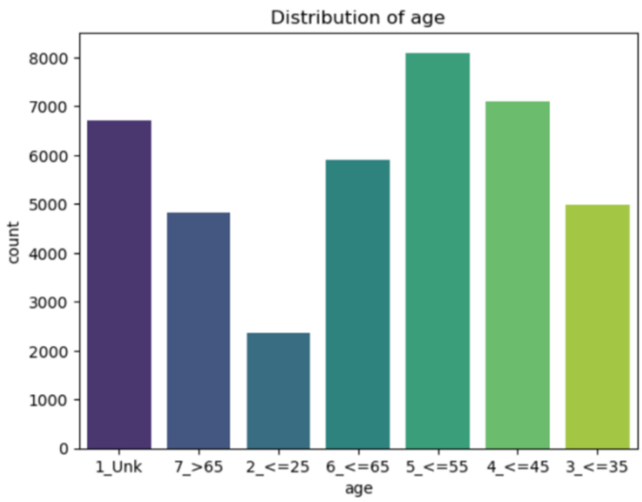
3.3.1 Exploratory Data Analysis (EDA)

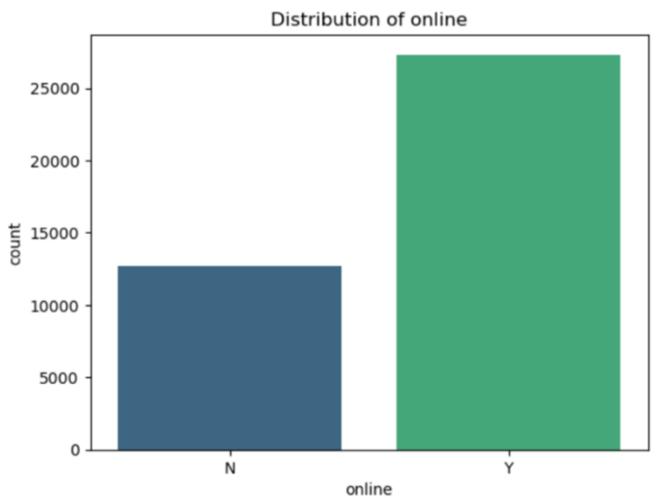
Exploratory Data Analysis commences with the examination of raw data in order to identify trends and patterns (James, 2023). The dataset provided for study by Imperials Ltd. encompasses a diverse range of variables that are essential for forecasting the purchase of life insurance products. The variables encompass demographic characteristics such as gender, age, and education; financial metrics like house worth and family income grade; as well as behavioural elements such as online purchase patterns and marital status. There are about 40,000 observations and 13 variables and flag (purchased) is our target Variable

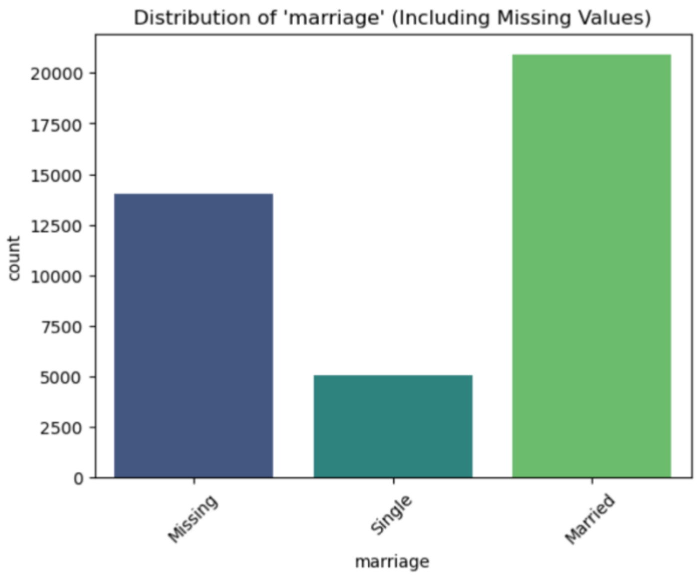
Data Variable	**Visualisation**	**Insights**
flag		The dataset used in this analysis has an equal distribution of the binary target variable 'flag,' with 'Y' (purchase made) and 'N' (no purchase) each accounting for 50% of the 40,000 records. This balanced distribution provides a solid basis for future predictive modelling efforts and eliminates the need for data imputation.

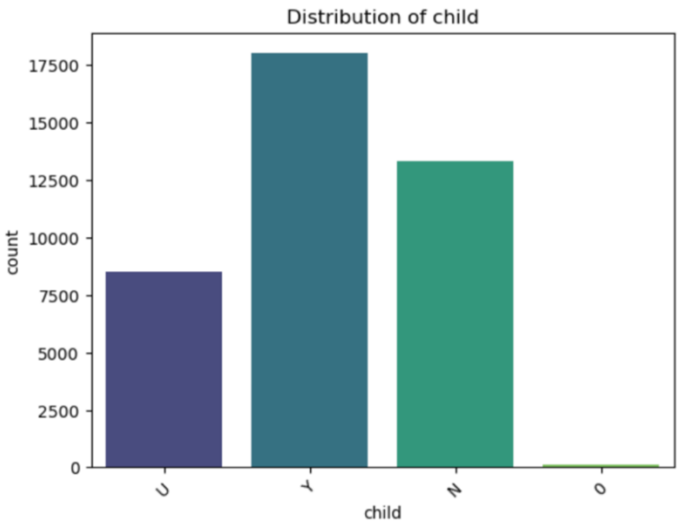
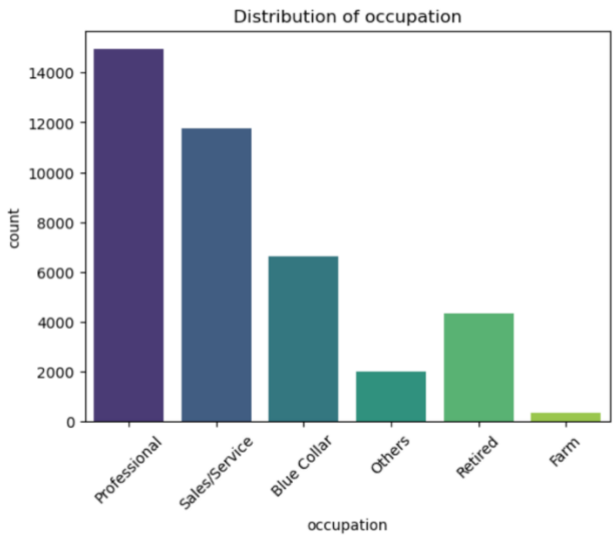
gender		The dataset includes 22,019 entries for males (M), 16,830 entries for females (F), and 1,151 items that are unidentified (U). This indicates a modest bias towards males, but there are no missing values.
education		The information indicates that the 'education' variable is primarily composed of those with a 'Some College' education level, followed by those who have completed high school ('HS'). A considerable proportion of the sample consists of persons who possess 'Bach' or bachelor's degrees, while a smaller but still significant number have postgraduate qualifications ('Grad'). There are a total of

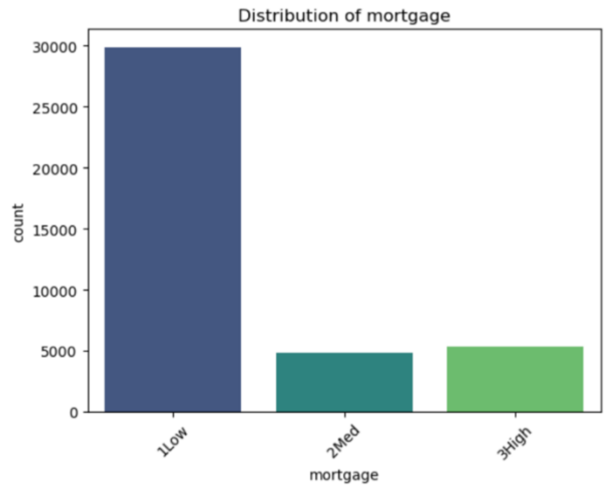
		741 entries that do not have educational data, which highlights the importance of carefully considering imputation procedures when using predictive modelling.
house_val		The dataset has a variable called 'house_val' which represents the values of houses. This variable shows a noticeable clustering of lower values, with a considerable number of houses having a value of zero. This indicates the presence of missing or placeholder data. The boxplot displays a broad spectrum of values, including prominent outliers that indicate attributes with

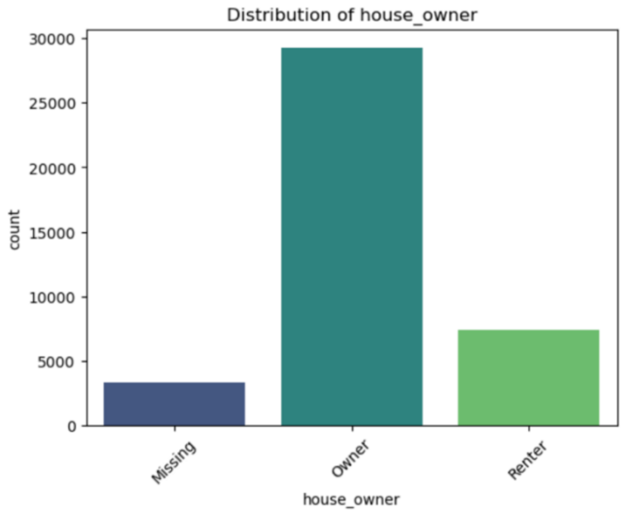
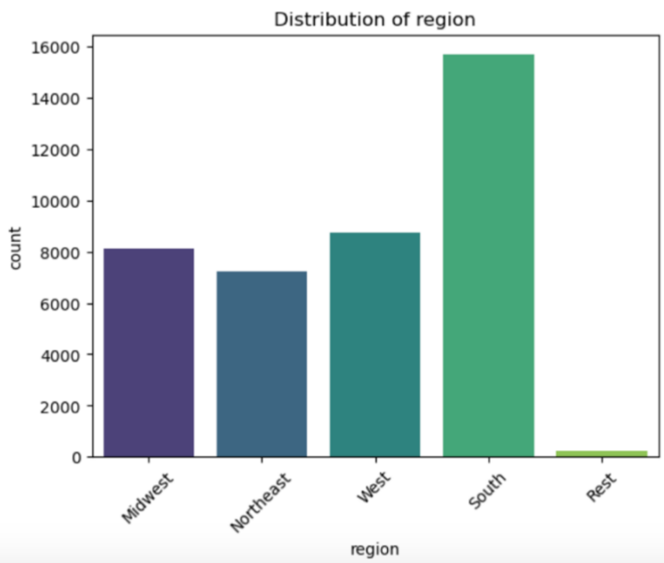
		unusually high values. Given the absence of any missing data, this financial measure is expected to be a significant factor in evaluating client categories and their likelihood to buy life insurance.
Age		The distribution of the 'age' variable exhibits a diverse demography, with the greatest number of individuals in the age group of 55 or younger, followed by the age group of 45 or younger. The age group of individuals aged 25 or below has the lowest number of people, while there is a significant portion of the population with undefined ages ('Unk'). The

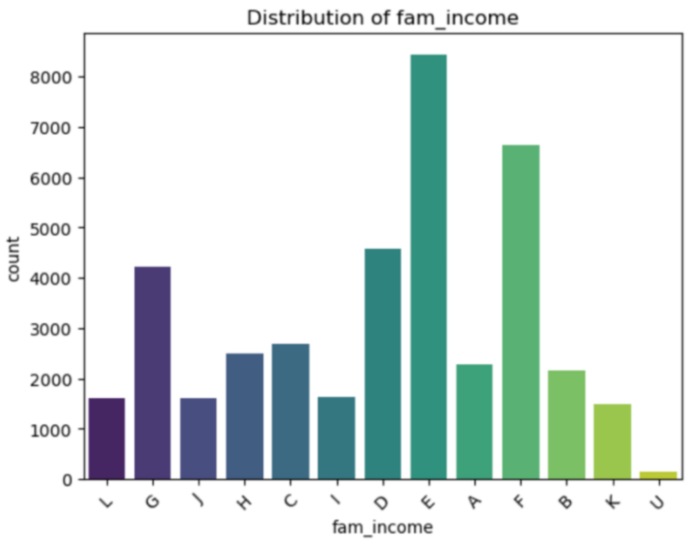
		distribution of this demographic is crucial for predictive modelling, as age frequently serves as a key indicator of insurance purchasing behaviour. This feature is fully complete and may now be included in analytics activities without any missing data.
online		The dataset includes a variable called 'online' that represents the online shopping behaviour of customers. Out of the total, 27,319 customers indicated 'Y' to indicate that they made purchases online, while 12,681 customers marked 'N' to indicate that they did not. The lack of missing values indicates that all data for this

		variable has been captured, which is crucial for comprehending the impact of online activity on purchasing decisions.
marriage		In the 'marriage' column, there are 20,891 individuals identified as married, 5,082 as single, and 14,027 entries are missing this information. The data showcases a higher proportion of married individuals within the dataset.

child		The dataset indicates that there are 18,012 customers with children, 13,333 customers without children, 8,528 entries that are unspecified (U), and 127 repetitions of the value '0', which suggest a data entry error. The presence of consumers with children in the dataset is a substantial component, which may have relevance in forecasting life insurance buying choices.
occupation		The 'occupation' variable categorization reveals the most represented group is 'Professional' with 14,936 entries, followed by 'Sales/Service' at 11,767, and 'Blue Collar' with 6,621 entries.

		'Retired' individuals count for 4,341, 'Others' are noted at 2,006, and the least represented are those listed under 'Farm' at 329, indicating a diverse occupational spread within the dataset, pivotal for analyzing purchasing behavior trends.
mortgage		The distribution for the 'mortgage' column indicates the majority of entries fall into the '1Low' mortgage bracket with 29,848 counts, suggesting most customers have lower mortgage obligations. The '3High' and '2Med' categories have significantly fewer representations with 5,349 and 4,803 counts respectively,

		revealing a dataset skewed towards lower mortgage brackets.
house_owner		The 'house_owner' column shows a total of 29,232 entries for homeowners and 7,391 entries for renters, with 3,377 missing values. Homeownership status is a larger segment within the dataset, which can be a significant variable in modeling insurance purchase propensity.
region		The data from the 'region' variable reveals that the South has the greatest client base, consisting of 15,676 persons. The West follows with 8,725 individuals, the Midwest with 8,107 individuals, and the Northeast with 7,247 individuals. The

		'Rest' category, consisting of 245 entries, indicates a dataset with a wide geographic dispersion. This is important for doing regional market analysis for life insurance products.
fam_income		The 'fam_income' category is distributed across a range from 'A' to 'L', signifying different income brackets, with 'E' being the most common, followed by 'F' and 'D'. The lowest income categories 'J', 'K', and 'L', as well as the 'U' category for unknown, have the fewest counts, depicting a dataset with a broad representation of family income levels, which could influence life insurance purchasing patterns.

3.3.2 Data quality Issues

After performing EDA we find the following data Quality Issues

Data Variable Name	Data quality issues
Gender	Upon observation, there are approximately 1151 instances of the letter 'U' that are not described in the data dictionary. Should be replaced these instances with 'male', since it is the most frequently occurring value (mode)
Education	The education variable has 741 missing values, which will be filled in using the mode of the education variable.
Marriage	The dataset contains missing values (NA) that need to be replaced with the mode.
Child	There is an undefined entry in the dataset, which consists of only 127 entries. This record can be eliminated.
House_owner	There are 3377 missing values (NA) that need to be replaced with the mode
house_val	There are outliers in the data that will be retained as they may represent houses with extremely high or low values and logically might be possible

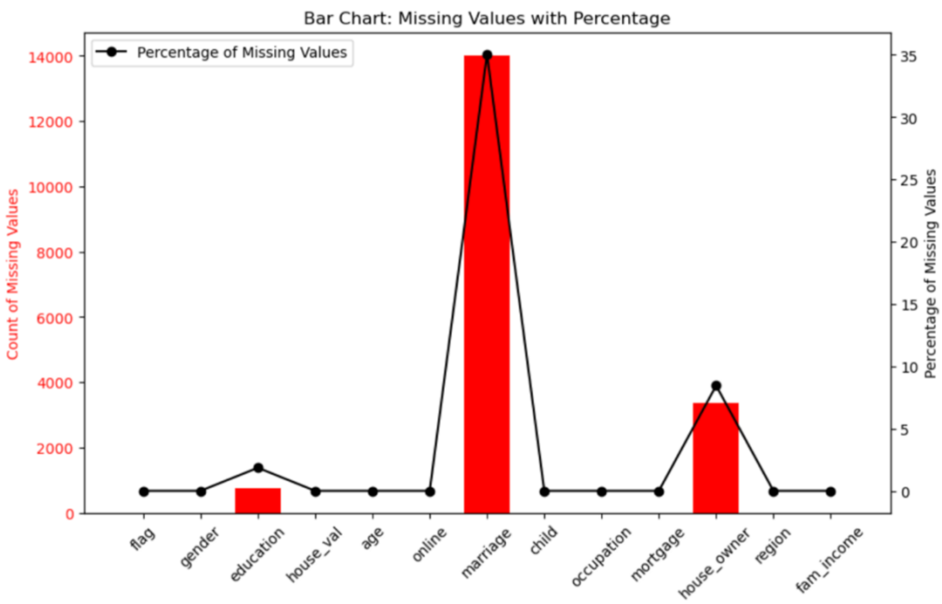


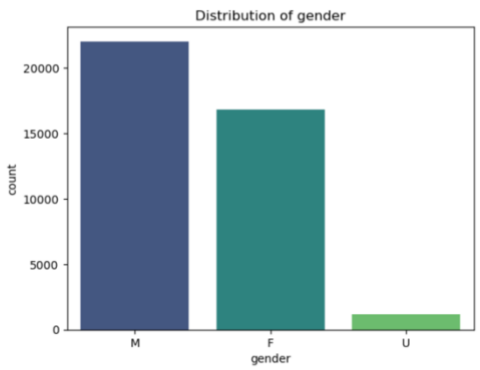
Fig 3.3.2: Bar plot of missing values

3.3.3 Addressing data quality issues

Data cleansing improves dataset integrity, essential before feature engineering tailored to the chosen model (Schröer et al., 2021). This process, critical for model optimization, hinges on the model's specifics, ensuring robust, actionable insight

1. **Gender**

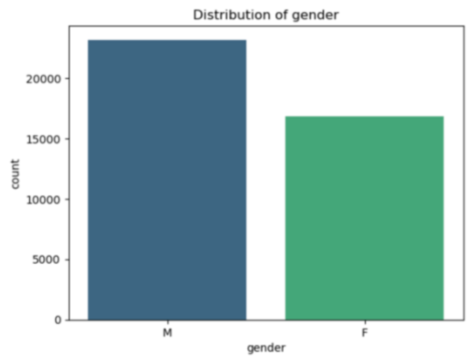
   The entry 'U' has be replaced by the mode i.e Male




```
The na present is  0
Summary of 'gender' column:
gender
M     22019
F     16830
U      1151
Name: count, dtype: int64
```

After

Cleaning

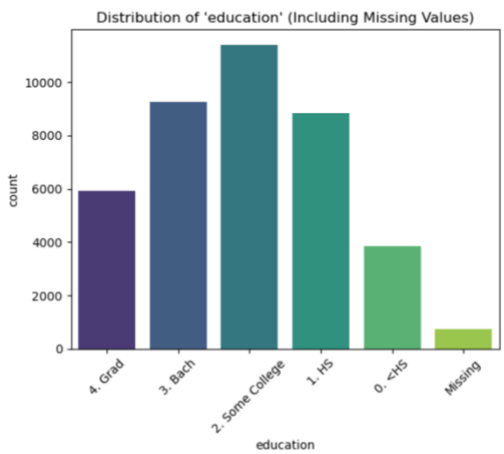



```
gender
M    23170
F    16830
Name: count, dtype: int64
```

Fig 3.3.3.1: Cleaning of Variable Gender

2. **Education**

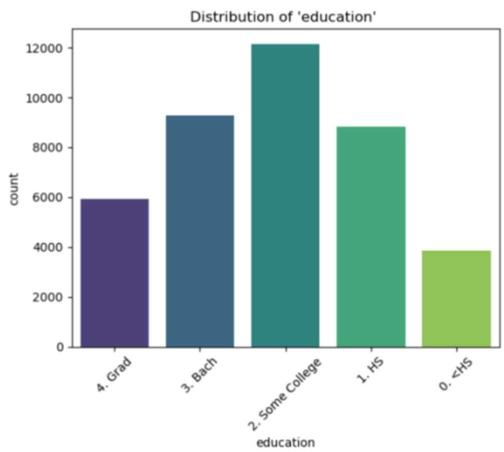
The education variable has 741 missing values, which has been imputed using the mode of the education variable.




```
The na present is  741
Summary of 'education' column:
education
2. Some College    11400
3. Bach             9267
1. HS               8828
4. Grad             5916
0. <HS              3848
Name: count, dtype: int64
```

After

Cleaning

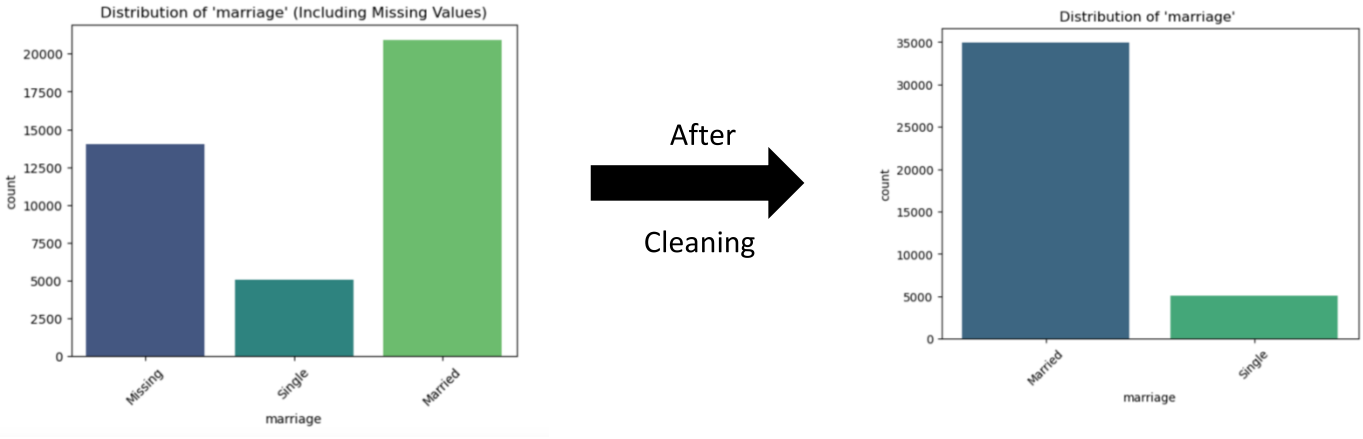



```
The na present is  0
Summary of 'education' column:
education
2. Some College    12141
3. Bach             9267
1. HS               8828
4. Grad             5916
0. <HS              3848
Name: count, dtype: int64
```

Fig 3.3.3.2: Cleaning of Variable Education

3. Marriage

The missing values are replace by mode i.e Instance Married



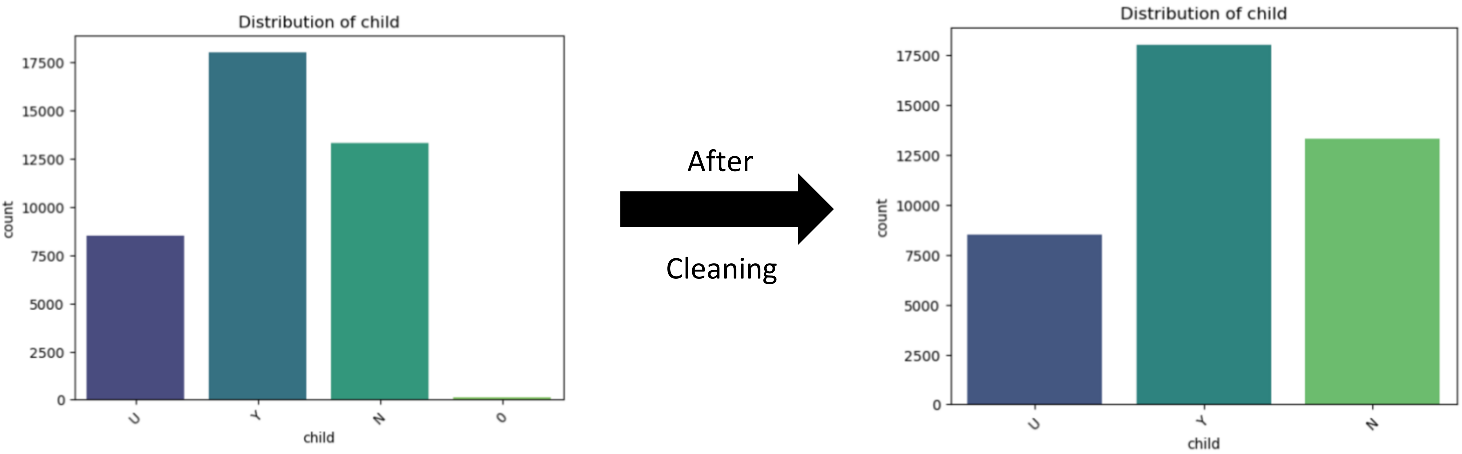

```
The na present is  14027
Summary of 'marriage' column:
marriage
Married    20891
Single      5082
Name: count, dtype: int64
```

```
Summary of 'marriage' column:
marriage
Married    34918
Single      5082
Name: count, dtype: int64
```

Fig 3.3.3.3: Cleaning of Variable Marriage

4. Child

There is an undefined entry in the dataset, which consists of only 127 entries. This record has been eliminated.



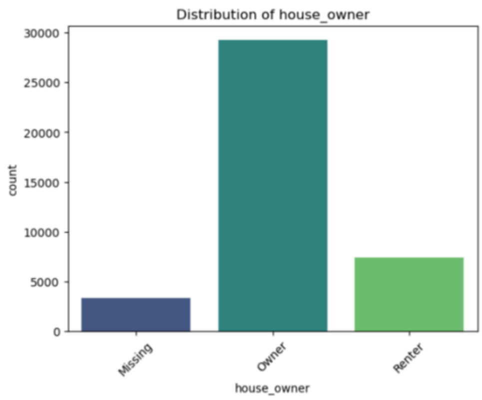

```
The na present is  0
Summary of 'child' column:
child
Y    18012
N    13333
U     8528
0      127
Name: count, dtype: int64
```

```
The na present is  0
Summary of 'child' column:
child
Y    18012
N    13333
U     8528
Name: count, dtype: int64
```

Fig 3.3.3.4: Cleaning of Variable child

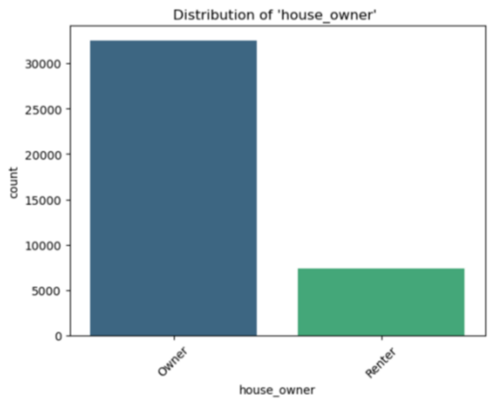
5. **House_owner**

There are 3377 missing values (NA) that has been to be replaced with mode.



After

Cleaning




```
The na present is  3377
Summary of 'house_owner' column:
house_owner
Owner      29232
Renter      7391
Name: count, dtype: int64
```

```
Summary of 'house_owner' column:
house_owner
Owner      32482
Renter      7391
Name: count, dtype: int64
```

Fig 3.3.3.5: Cleaning of Variable House_owner

4.0 Data Analytics

4.1 Descriptive Statistics and Visualisations

4.1.1 Correlation

Chi square Test

The chi-square (χ2) test is a non-parametric statistical technique used to determine if significant differences exist between observed and expected frequencies in nominal data categories. Its importance lies in its robustness to violations of parametric assumptions, ability to analyze group differences for nominal variables, and provision of detailed information about group performances, enhancing result interpretation (McHugh, 2013).

4.1.2 Insightful Visualisations and correlation of variables

Based on the literature review we have found 5 variables which might have impact on our target variable ***"flag (purchased)"***

1. **Age vs Flag (purchased)** (Mau et al., 2018)

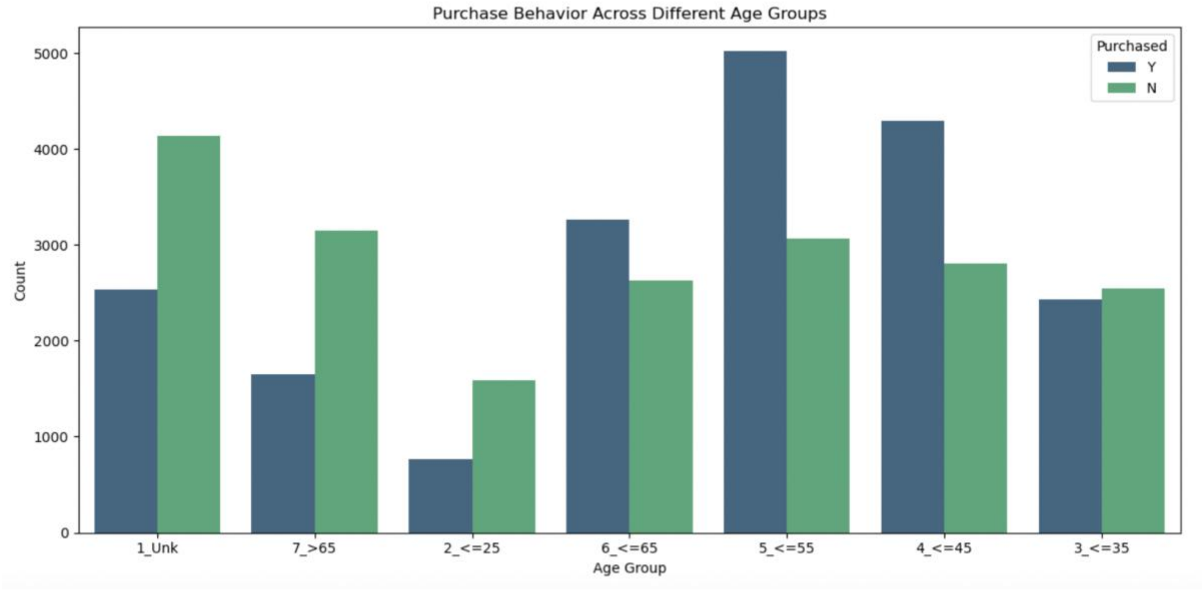


Fig 4.1.2.1: Age vs Flag (Purchased)

```
Chi-square Statistic: 1993.6352210016457, p-value: 0.0
Degrees of Freedom: 6
P-value: 0.0
Expected Frequencies: [[3334.24302159 3337.75697841]
 [1175.38063351 1176.61936649]
 [2486.68963961 2489.31036039]
 [3548.13031375 3551.86968625]
 [4040.87066436 4045.12933564]
 [2942.44947709 2945.55052291]
 [2398.23625009 2400.76374991]]
```

Fig 4.1.2.2: Age vs Flag Chi-sq test

The bar chart shows purchasing habits by age cohort, comparing those who bought ("Y") to those who didn't ("N"). The 1993.6352 statistic and 0.0 p-value of the chi-square test show a statistically significant association between age groups and purchase habits. . Additionally, the projected frequencies differ significantly from the observed counts, emphasising the differences. The age cohorts 6_=65 and 4_=45 are more likely to buy, which can help build targeted marketing tactics.

2. Gender vs Flag (purchased) (Scriney et al., 2020)

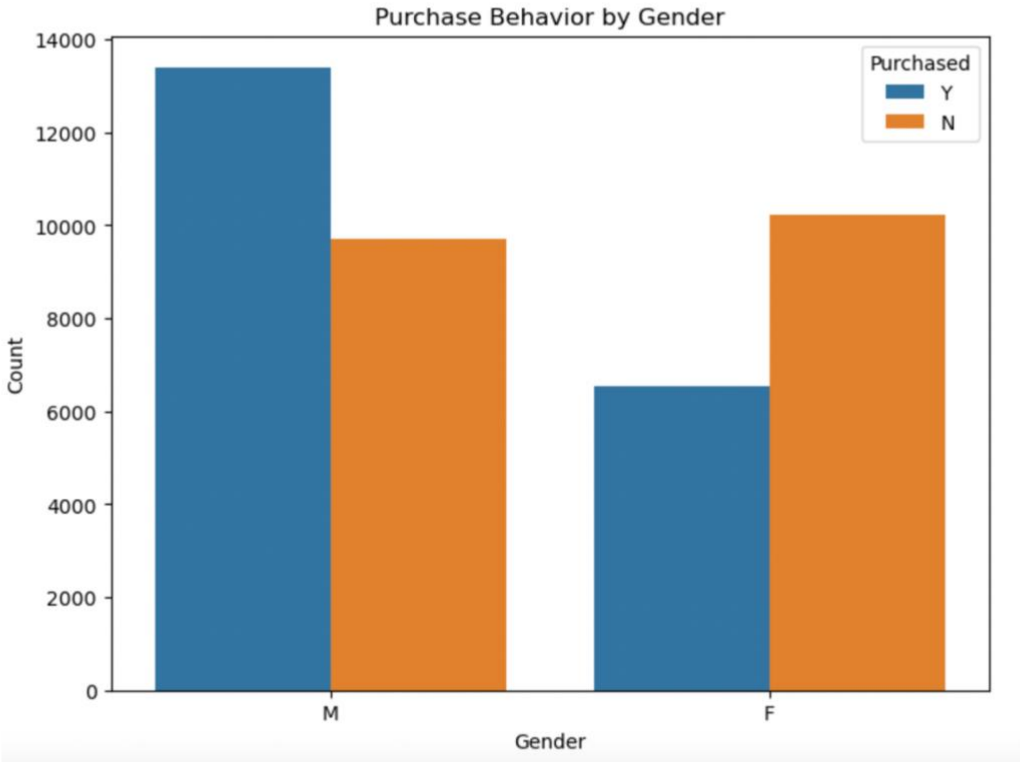


Fig 4.1.2.3: Gender vs Flag (Purchased)

```
Chi-square Statistic: 1399.0049840345037, p-value: 3.456592567593227e-306
Degrees of Freedom: 1
P-value: 3.456592567593227e-306
Expected Frequencies: [[ 8378.08517042  8386.91482958]
 [11547.91482958 11560.08517042]]
```

Fig 4.1.2.4: Gender vs Flag Chi-sq Test

The bar chart illustrates purchase behavior by gender (Males: M, Females: F), comparing those who bought ("Y") and those who didn't ("N"). The chi-square test reveals a highly significant difference in buying behavior between genders ($p \approx 3.46e-306$, $\chi^2 = 1399.005$), indicating statistical significance. Anticipated frequencies assuming no gender disparity significantly differ from observed counts. This suggests gender as a determinant of purchasing behavior, with one gender showing a notably higher inclination to purchase compared to the other.

3. Education level vs Flag (purchased) (Boodhun & Jayabalan, 2018)

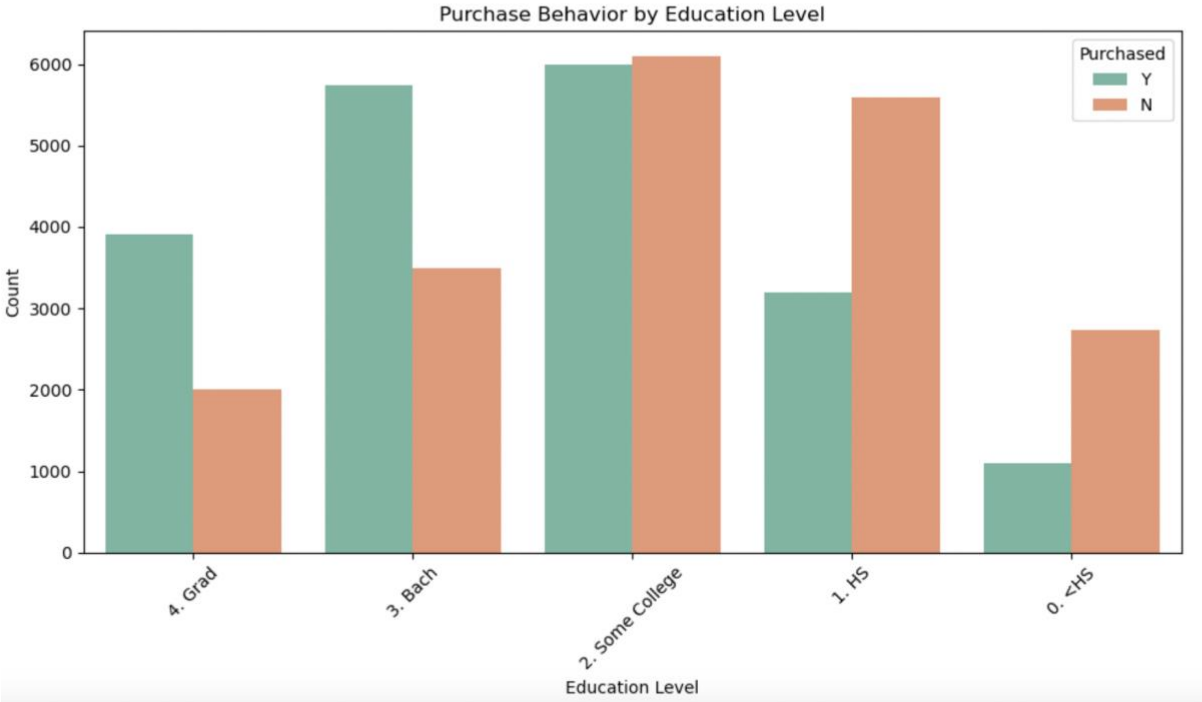


Fig 4.1.2.5: Education level vs Flag (Purchased)

```
Chi-square Statistic: 2501.635544132398, p-value: 0.0
Degrees of Freedom: 4
P-value: 0.0
Expected Frequencies: [[1916.49010609 1918.50989391]
 [4395.68369573 4400.31630427]
 [6041.81626665 6048.18373335]
 [4621.06493116 4625.93506884]
 [2950.94500038 2954.05499962]]
```

Fig 4.1.2.6: Education level vs Flag Chi-sq Test

The bar chart reveals a strong correlation between educational attainment and purchasing behavior, with statistically significant findings. Higher education levels, especially bachelor's degrees, correspond to increased buying activity, while lower education levels show fewer purchases. The high chi-square statistic and near-zero p-value validate this pattern's significance, highlighting education's notable influence on consumer behavior.

4. Marriage vs flag (purchased) (Mau, Pletikosa, and Wagner, 2018)

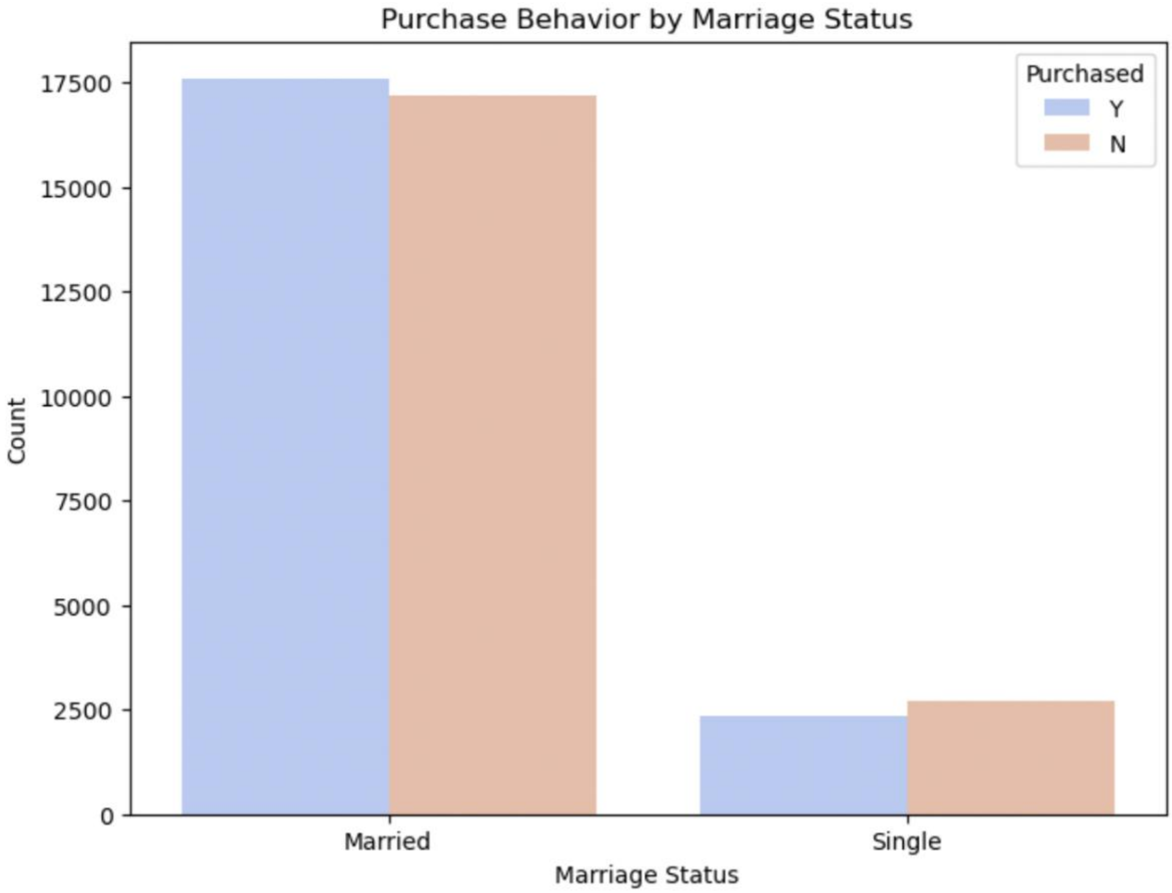


Fig 4.1.2.7: Marriage vs Flag (Purchased)

```
Chi-square Statistic: 33.80274216013279, p-value: 6.099283134621229e-09
Degrees of Freedom: 1
P-value: 6.099283134621229e-09
Expected Frequencies: [[17393.83432398 17412.16567602]
 [ 2532.16567602  2534.83432398]]
```

Fig 4.1.2.8: Marriage vs Flag Chi-sq Test

The bar chart visually compares purchase behavior across marital statuses, indicating married individuals engage more in purchasing activities. A chi-square test confirms a significant association ($\chi^2 = 33.80$, $p \approx 6.10e\text{-}09$), suggesting marital status strongly influences buying behavior. Observed frequencies significantly differ from expected values, emphasizing marital status as a crucial factor in consumer behavior.

5. Occupation vs flag (purchased)

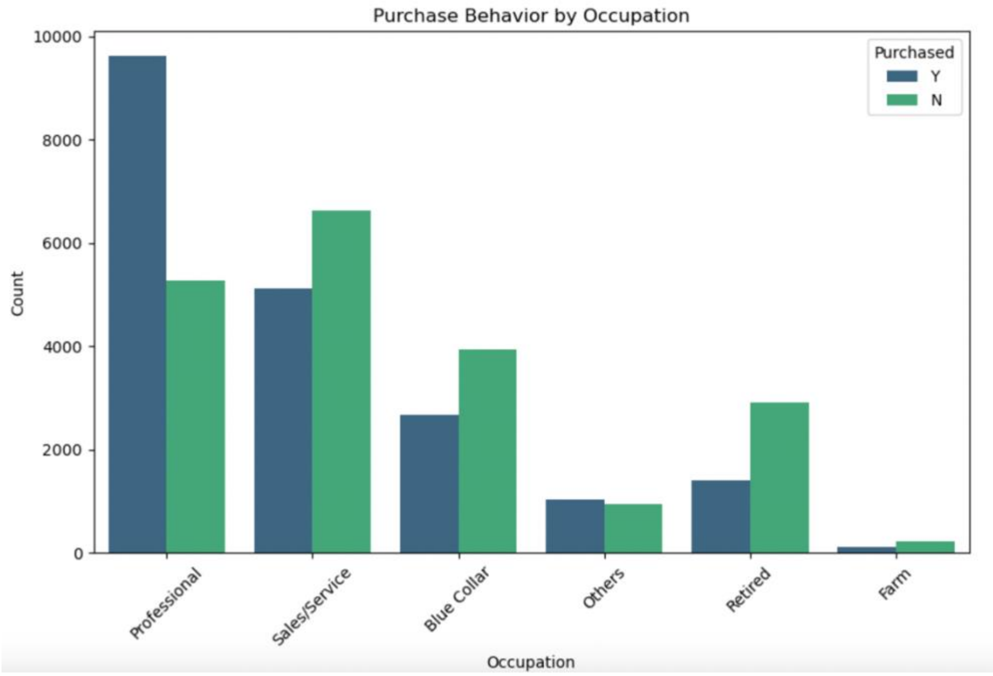


Fig 4.1.2.9: Occupation vs Flag (Purchased)

```
Chi-square Statistic: 2268.046105792107, p-value: 0.0
Degrees of Freedom: 5
P-value: 0.0
Expected Frequencies: [[3301.26040178 3304.73959822]
 [ 164.41336243  164.58663757]
 [ 990.97780453  992.02219547]
 [7445.57655556 7453.42344444]
 [2160.86133474 2163.13866526]
 [5862.91054097 5869.08945903]]
```

Fig 4.1.2.10: Occupation vs Flag Chi-sq Test

The bar chart depicts purchasing patterns across various professions, with professionals as the largest purchasers and agriculture as the smallest. Sales/service and blue-collar workers also contribute significantly. Chi-square test results ($\chi^2 = 2268.046$, $p = 0.0$) indicate a significant difference in purchase behavior among occupational categories, suggesting occupation strongly predicts buying behavior. Anticipated frequencies, differing notably from observed counts, support this, ruling out random variation.

4.2 Supervised Machine Learning

We have chosen two machine learning models.

1. **Logistic Regression**
   Logistic regression is a statistical method used for predicting binary outcomes, utilizing both continuous and categorical variables. It shines in analyzing observational data, adjusting for multiple predictors to minimize bias. Its ability to provide odds ratios makes it particularly effective for understanding the impact of various factors on dichotomous outcomes, making it indispensable in fields such as medicine and social sciences (LaValley, M. P., 2008).

2. **Random Forest**
   Random Forest is a machine learning technique that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. It offers significant accuracy in predictive models by utilizing bootstrap aggregation and randomization of predictors, making it especially powerful in the era of "big data" for uncovering complex relationships and interactions among variables without specifying them in advance (Rigatti, 2017).

4.2.1 Data splitting methodology

In preparing the dataset for analysis, object-type variables were systematically transformed into categorical variables to optimize algorithmic performance. Subsequently, the dataset was divided, employing a reproducible method that allocated a 70% subset for model training and 30% for validation, with the partitioning process governed by a predetermined random seed to maintain consistency.

4.2.2 Model Training and Selection

Scaling and standardisation

The dataset was preprocessed using a `ColumnTransformer`, which standardized numerical features and applied one-hot encoding to categorical features. Standardization ensures that numerical data have a consistent scale, while encoding transforms categorical variables into a format suitable for machine learning models. This preprocessing was applied to both the training and testing sets to maintain uniformity.

Hyper Parameter Training

Hyperparameter optimization is crucial in the development of machine learning models as it directly influences a model's performance by selecting the best hyperparameter configuration. This process not only reduces the manual effort involved in tuning, which is particularly significant for complex models with many hyperparameters but also enhances the model's performance across different datasets by finding dataset-specific optimums. Moreover, hyperparameter optimization contributes to the reproducibility of models and research findings. By applying a consistent hyperparameter tuning process, various machine learning algorithms can be compared fairly, assisting in identifying the most suitable model for a given problem (Yang & Shami, 2020).

4.2.2.1 Logistic Regression model with parameter training

The logistic regression classifier was chosen for binary classification, using a regularization strength (C) of 1.0 for balance between minimizing error and generalization. A high iteration limit of 10,000 ensured model convergence. The training involved preprocessing data to find distinguishing patterns, with crucial hyperparameters C and max_iter influencing decision complexity and computational effort. Fine-tuning these parameters resulted in an optimized classifier tailored to the dataset's specific needs.

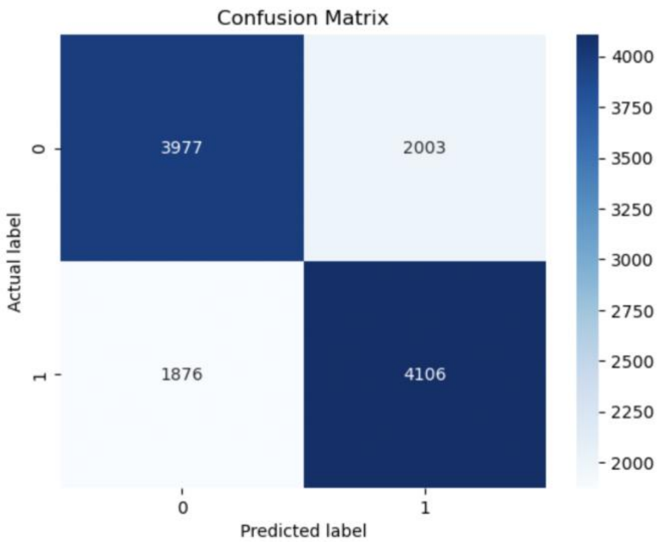
4.2.2.2 Random Forest with parameter training

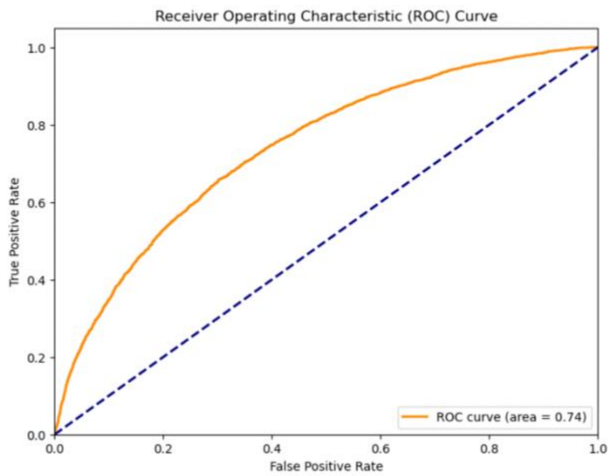
The Random Forest classifier was chosen for its ability to minimize overfitting and enhance accuracy by averaging multiple decision trees' outcomes. Hyperparameter tuning was conducted using grid search and cross-validation, focusing on ensemble size (n_estimators), maximum tree depth (max_depth), and minimum samples for node splitting (min_samples_split). This approach ensured the selection of the optimal model configuration, aiming to improve the algorithm's predictive performance and robustness across varied data.

4.3 Model evaluation and comparison

4.3.1 Logistic regression output

The logistic regression model demonstrated approximately 67.57% accuracy, precision, recall, and F1 score, with a confusion matrix revealing 3977 true negatives and 4106 true positives, in addition to 2003 false positives and 1876 false negatives. An AUC of 0.74 indicates the model's moderate ability in class differentiation, pointing to its satisfactory performance in current evaluations.






```
Accuracy: 0.6757
Precision: 0.6758
Recall: 0.6757
F1 Score: 0.6757
```

Fig 4.3.1.: Logistic Regression Output

4.3.2 Random Forest model output

The Random Forest classifier achieved a precision of 68.10%, with similar precision and recall rates indicating its reliable prediction capability. An F1 score of 68.09% reflects a balanced precision-recall trade-off. The model identified 3995 true negatives and 4151 true positives, with 1985 false positives and 1831 false negatives. An AUC of 0.74 confirms its satisfactory class differentiation, showcasing its efficacy as a predictive tool.

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
Best Random Forest Metrics:
Accuracy: 0.6810
Precision: 0.6811
Recall: 0.6810
F1 Score: 0.6809
```

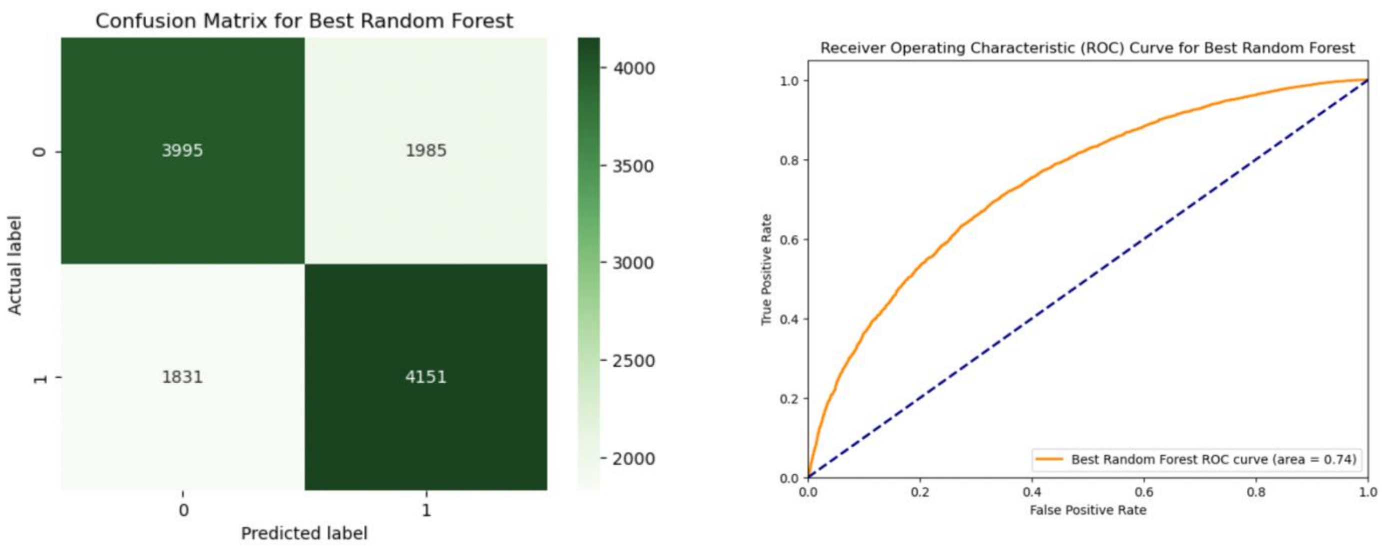


Fig 4.3.2.: Random Forest Model Output

4.4 Comparison of models

The Random Forest classifier slightly outperforms the logistic regression model across accuracy, precision, recall, and F1 score by a narrow margin of approximately 0.05%. This suggests Random Forest's superior ability to discern and utilize underlying data patterns, likely owing to its ensemble method that combines multiple decision trees to enhance prediction accuracy while minimizing overfitting. Analysis of the confusion matrices indicates that Random Forest not only reduces false negatives but also marginally increases true positives and negatives, underscoring its enhanced capability in accurately classifying instances across both classes. Both models achieved an identical Area Under the Curve (AUC) value of 0.74, signifying a comparable classification performance, especially in ranking positive instances over negative ones.

5.0 Results And discussion

The analysis conducted on the dataset from Imperials Ltd aimed at predicting which consumers are more likely to purchase life insurance. This study utilized logistic regression and Random Forest models due to their proven efficacy in similar predictive analytics tasks. The logistic regression model exhibited an accuracy of approximately 67.57%, with a precision, recall, and F1 score of similar values, and an AUC of 0.74. Meanwhile, the Random Forest model slightly outperformed the logistic regression with an accuracy, precision, recall, and F1 score of around 68.10% and an identical AUC of 0.74.

The marginal superiority of the Random Forest model suggests its better capability to harness and interpret the underlying patterns within the dataset, potentially due to its ensemble approach that aggregates predictions from multiple decision trees to improve accuracy and reduce overfitting. Despite this, both models showed comparable performance, indicating that either could be effectively employed for the predictive task at hand, with Random Forest offering a slight edge in accuracy and reliability.

Throughout the analysis, several challenges and limitations were encountered, including the handling of missing data, the need for feature engineering to improve model performance, and the balancing of the dataset to prevent model bias. These challenges underscore the importance of rigorous data pre-processing and the strategic selection of machine learning models tailored to the specific characteristics of the data.

6.0 Conclusion and recommendations

The study's findings are significant for Imperials Ltd's strategic objectives of enhancing life insurance sales through predictive analytics. The slight edge of the Random Forest model in accuracy and the challenges encountered during the analysis provide valuable insights into the nuances of predictive modeling in the insurance domain.

Key Findings:

The Random Forest model's marginally better performance highlights the importance of ensemble methods in dealing with complex datasets.

The importance of rigorous data pre-processing and feature engineering is underscored by the challenges encountered, which include addressing missing data and ensuring data quality.

Recommendations

For Imperials Ltd, adopting the Random Forest model is recommended for its accuracy in predicting life insurance purchases. Enhancing data quality is crucial for the improved performance of predictive models, alongside exploring advanced machine learning techniques to gain deeper insights into customer behavior. Implementing real-time analytics can enable swift adjustments to marketing strategies, capitalizing on emerging opportunities. Additionally, systematically gathering and analyzing customer feedback will refine models and marketing strategies, ensuring they remain relevant and effective. Cultivating a data-driven culture within the organization ensures decision-making at all levels is informed by analytical insights, positioning Imperials Ltd for greater success in its predictive analytics endeavors.

7.0 References

- Abedi, R. et al. (2021) 'Flash-flood susceptibility mapping based on XGBoost, random forest and boosted regression trees', Geocarto International, 37(19), pp. 5479–5496. doi:10.1080/10106049.2021.1920636 Available at https://www.tandfonline.com/doi/full/10.1080/10106049.2021.1920636 (Accessed at 4th March 2023).

- Boodhun, N. and Jayabalan, M. (2018) 'Risk prediction in life insurance industry using supervised learning algorithms', Complex & Intelligent Systems, 4(2), pp. 145–154. doi:Available at: https://link.springer.com/article/10.1007/s40747-018-0072-1 [Accessed on march 2nd 2024].

- Fang, K., Jiang, Y. and Song, M. (2016) 'Customer profitability forecasting using Big Data Analytics: A case study of the insurance industry', Computers & Industrial Engineering, 101, pp. 554–564. doi:10.1016/j.cie.2016.09.011 Available at https://www.sciencedirect.com/science/article/pii/S0360835216303515 (Accessed on 9th March 2023).

- Guleria, P. and Sood, M. (2018a) 'Predictive data modeling: Educational data classification and comparative analysis of classifiers using Python', 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) [Preprint]. doi:10.1109/pdgc.2018.8745727 Available at https://ieeexplore.ieee.org/abstract/document/8745727 (Accessed on 10th March 2023).

- Hand, D.J. (2001) 'Modelling consumer credit risk', IMA Journal of Management Mathematics, 12(2), pp. 139–155. doi:Available at : https://academic.oup.com/imaman/article-abstract/12/2/139/750622 [Accessed on march 2nd 2024].

- James, G. et al. (2023) An introduction to statistical learning with applications in Python. Cham: Springer International Publishing.

- Kumar, M., Ghani, R. and Mei, Z.-S. (2010) 'Data mining to predict and prevent errors in health insurance claims processing', Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining [Preprint]. doi:10.1145/1835804.1835816 Available at https://dl.acm.org/doi/abs/10.1145/1835804.1835816 Accessed on 12th March 2024.

- LaValley, M.P. (2008) 'Logistic regression', Circulation, 117(18), pp. 2395–2399. doi:Available at : https://www.ahajournals.org/doi/epub/10.1161/CIRCULATIONAHA.106.682658 [Accessed on march 2nd 2024].

- Lipinski, P., Brzychczy, E. and Zimroz, R. (2020) ‘Decision tree-based classification for planetary gearboxes’ condition monitoring with the use of vibration data in multidimensional symptom space’, Sensors, 20(21), p. 5979. doi:10.3390/s20215979 Available at https://www.mdpi.com/1424-8220/20/21/5979 (Accessed on 10th March 2023).

- Marbn, O., Mariscal, G. and Segovi, J. (2009) ‘A Data Mining & Knowledge Discovery Process Model’, Data Mining and Knowledge Discovery in Real Life Applications [Preprint]. doi:10.5772/6438 Available at https://cdn.intechopen.com/pdfs/5937/InTech-A/_data/_mining/_amp/_knowledge/_discovery/_process/_model.pdf (Accessed on 5th March 2023).

- Mau, S., Pletikosa, I. and Wagner, J. (2018) ‘Forecasting the next likely purchase events of insurance customers’, International Journal of Bank Marketing, 36(6), pp. 1125–1144. doi:Available at : https://www.emerald.com/insight/content/doi/10.1108/IJBM-11-2016-0180/full/html?skipTracking=true [Accessed on march 2nd 2024].

- Rao, H., Shi, X., Rodrigue, A.K., et al. (2019) ‘Feature selection based on artificial bee colony and Gradient Boosting Decision tree’, Applied Soft Computing, 74, pp. 634–642. doi:10.1016/j.asoc.2018.10.036.

- Rigatti, S.J. (2017) ‘Random Forest’, Journal of Insurance Medicine, 47(1), pp. 31–39. doi:Available at :https://meridian.allenpress.com/jim/article/47/1/31/131479/Random-Forest [Accessed on march 2nd 2024].

- Sawant, N. and Khadapkar, D.R. (2022) ‘Comparison of the performance of gaussiannb algorithm, the k neighbors classifier algorithm, the logistic regression algorithm, the linear discriminant analysis algorithm, and the decision tree classifier algorithm on same dataset’, International Journal for Research in Applied Science and Engineering Technology, 10(12), pp. 1654–1665. doi:10.22214/ijraset.2022.48311.

- Schröer, C., Kruse, F. and Gómez, J.M. (2021) ‘A systematic literature review on applying CRISP-DM process model’, Procedia Computer Science, 181, pp. 526–534. doi:10.1016/j.procs.2021.01.199 Available at https://www.sciencedirect.com/science/article/pii/S1877050921002416 (Accessed at 4th March 2023).

- Scriney, M., Nie, D. and Roantree, M. (2020) ‘Predicting customer churn for insurance data’, Big Data Analytics and Knowledge Discovery, pp. 256–265. doi:Available at : https://link.springer.com/chapter/10.1007/978-3-030-59065-9_21#citeas [Accessed on march 2nd 2024].

- Shi, P. and Shi, K. (2022) ‘Non-life insurance risk classification using categorical embedding’, North American Actuarial Journal, 27(3), pp. 579–601. doi:10.1080/10920277.2022.2123361 Available at
- https://www.tandfonline.com/doi/full/10.1080/10920277.2022.2123361 (Accessed at 5th March 2023).

- Shamsuddin, S.N., Ismail, N. and Nur-Firyal, R. (2023) ‘Life insurance prediction and its sustainability using machine learning approach’, Sustainability, 15(13), p. 10737. doi:Available at : https://www.mdpi.com/2071-1050/15/13/10737 [Accessed on march 2nd 2024].

- Song, I.J. and Heo, W. (2022) ‘Improving insurers’ loss reserve error prediction: Adopting combined unsupervised-supervised machine learning techniques in Risk Management’, The Journal of Finance and Data Science, 8, pp. 233–254. doi:10.1016/j.jfds.2022.09.003 Available at https://www.sciencedirect.com/science/article/pii/S2405918822000137 (Accessed on 5th March 2023).

- Speiser, J.L. et al. (2019) ‘A comparison of random forest variable selection methods for classification prediction modeling’, Expert Systems with Applications, 134, pp. 93–101. doi:10.1016/j.eswa.2019.05.028 Available at https://pdf.sciencedirectassets.com/271506 [Accessed on - 5th March 2023.]

- Subudhi, S. and Panigrahi, S. (2018) ‘Detection of automobile insurance fraud using feature selection and data mining techniques’, International Journal of Rough Sets and Data Analysis, 5(3), pp. 1–20. doi:10.4018/ijrsda.2018070101 Available at https://www.aeuso.org/includes/files/articles/Vol8_Iss27_3764-3771_Fraud_Detection_in_Automobile_Insur.pdf [Accessed on 11th March 2023].

- Yang, L. and Shami, A. (2020) ‘On hyperparameter optimization of Machine Learning Algorithms: Theory and practice’, Neurocomputing, 415, pp. 295–316. doi:10.1016/j.neucom.2020.07.061 Available at https://pdf.sciencedirectassets.com/271597/1-s2.0 [Accessed on 15th March 2023].

- James, G. et al. (2023) An introduction to statistical learning with applications in Python. Cham: Springer International Publishing.

8.0 appendix

Python code

```
sur#!/usr/bin/env python
# coding: utf-8


# In[39]:



import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import chi2_contingency
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, roc_curve, auc
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('/Users/dhanush/Desktop/Bussiness analytics /Sem 2/Data
Mining/sales_data (1).csv')

# Display the first few rows of the DataFrame to verify it's loaded correctly
print(data.head())

#print all the column names
print(data.columns)
```

```
# In[2]:


#Data exploration

#1)Flag

#Count missing values in the 'flag' column
flag_na =data['flag'].isna().sum()
print("The na present is ",flag_na)


# Provide a summary of the 'flag' column (count of each unique value)
flag_summary = data['flag'].value_counts()
print("Summary of 'flag' column:")
print(flag_summary)

# 4. Create a bar plot showing the distribution of values in the 'flag' column
sns.countplot(x='flag', data=data, palette='viridis')
plt.title("Distribution of 'flag' Values")
plt.show()


# In[3]:


#2)gender
```

```
#Count missing values in the 'Gender' column
gender_na = data['gender'].isna().sum()
print("The na present is ",gender_na)

# Provide a summary of the 'Gender' column (count of each unique value)
age_summary = data['gender'].value_counts()
print("Summary of 'gender' column:")
print(age_summary)

# 4. Create a bar plot showing the distribution of values in the 'gender' column
sns.countplot(x='gender', data=data, palette='viridis')
plt.title("Distribution of gender")
plt.show()

#there is some unkown u in the data which has to removed or imputed



# In[4]:



#3)education

#Count missing values in the 'education' column
edu_na = data['education'].isna().sum()
print("The na present is ",edu_na)



# Provide a summary of the 'education' column (count of each unique value)
education_summary = data['education'].value_counts()
print("Summary of 'education' column:")
```

```
print(education_summary)

# 4. Create a bar plot showing the distribution of values in the 'education' column
# Fill NA values with a placeholder for 'education'
data_filled_education = data.copy()
data_filled_education['education'] = data_filled_education['education'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'education' column,
including NA values
sns.countplot(x='education', data=data_filled_education, palette='viridis')
plt.title("Distribution of 'education' (Including Missing Values)")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()



#there are so na which has to be removed



# In[5]:



#4)house_val

#Count missing values in the 'house_val' column
house_val_na = data['house_val'].isna().sum()
print("The na present is ",house_val_na)


# Provide a summary of the 'house_val' column (count of each unique value)
house_val_summary = data['house_val'].value_counts()
```

```
print("Summary of 'house_val' column:")
print(house_val_summary)

# 4. Create a box plot showing the distribution of values in the 'house_val' column
sns.boxplot(x=data['house_val'])
plt.title("Boxplot of House Values")
plt.xlabel("House Value")
plt.show()

# need to normalise the data



# In[6]:



#5)age

#Count missing values in the 'age' column
age_na = data['age'].isna().sum()
print("The na present is ",age_na)



# Provide a summary of the 'age' column (count of each unique value)
age_summary = data['age'].value_counts()
print("Summary of 'age' column:")
print(age_summary)

# 4. Create a bar plot showing the distribution of values in the 'education' column
sns.countplot(x='age', data=data, palette='viridis')
plt.title("Distribution of age")
```

```
plt.show()


# In[7]:


#6)online

#Count missing values in the 'online' column
online_na = data['online'].isna().sum()
print("The na present is ",online_na)


# Provide a summary of the 'online' column (count of each unique value)
online_summary = data['online'].value_counts()
print("Summary of 'online' column:")
print(online_summary)

# 4. Create a bar plot showing the distribution of values in the 'online' column
sns.countplot(x='online', data=data, palette='viridis')
plt.title("Distribution of online")
plt.show()


# In[8]:


#7)marriage

#Count missing values in the 'marriage' column
```

```
marriage_na = data['marriage'].isna().sum()
print("The na present is ",marriage_na)



# Provide a summary of the 'marriage' column (count of each unique value)
marriage_summary = data['marriage'].value_counts()
print("Summary of 'marriage' column:")
print(marriage_summary)

# 4. Create a bar plot showing the distribution of values in the 'marriage' column
data_filled = data.copy()
data_filled['marriage'] = data_filled['marriage'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'marriage' column, including NA values
sns.countplot(x='marriage', data=data_filled, palette='viridis')
plt.title("Distribution of 'marriage' (Including Missing Values)")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()



# In[9]:



#8) child

#Count missing values in the 'child' column
child_na = data['child'].isna().sum()
print("The na present is ",child_na)
```

```
# Provide a summary of the 'child' column (count of each unique value)
child_summary = data['child'].value_counts()
print("Summary of 'child' column:")
print(child_summary)

# 4. Create a bar plot showing the distribution of values in the 'child' column
data_filled = data.copy()
data_filled['child'] = data_filled['child'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'child' column, including
NA values
sns.countplot(x='child', data=data_filled, palette='viridis')
plt.title("Distribution of child")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()


# In[10]:


#9) occupation

#Count missing values in the 'occupation' column
occupation_na = data['occupation'].isna().sum()
print("The na present is ",occupation_na)
```

```
# Provide a summary of the 'occupation' column (count of each unique value)
occupation_summary = data['occupation'].value_counts()
print("Summary of 'occupation' column:")
print(occupation_summary)

# 4. Create a bar plot showing the distribution of values in the 'occupation' column
data_filled = data.copy()
data_filled['occupation'] = data_filled['occupation'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'occupation' column,
including NA values
sns.countplot(x='occupation', data=data_filled, palette='viridis')
plt.title("Distribution of occupation")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()




# In[11]:



#10) mortgage

#Count missing values in the 'mortgage' column
mortgage_na = data['mortgage'].isna().sum()
print("The na present is ",mortgage_na)



# Provide a summary of the 'mortgage' column (count of each unique value)
mortgage_summary = data['mortgage'].value_counts()
```

```
print("Summary of 'mortgage' column:")
print(mortgage_summary)

# 4. Create a bar plot showing the distribution of values in the 'occupation' column
data_filled = data.copy()
data_filled['mortgage'] = data_filled['mortgage'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'occupation' column,
including NA values
sns.countplot(x='mortgage', data=data_filled, palette='viridis')
plt.title("Distribution of mortgage")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()



# In[12]:


#11) house_owner

#Count missing values in the 'house_owner' column
house_owner_na = data['house_owner'].isna().sum()
print("The na present is ",house_owner_na)


# Provide a summary of the 'mortgage' column (count of each unique value)
house_owner_summary = data['house_owner'].value_counts()
print("Summary of 'house_owner' column:")
print(house_owner_summary)
```

```
# 4. Create a bar plot showing the distribution of values in the 'house_owner' column
data_filled = data.copy()
data_filled['house_owner'] = data_filled['house_owner'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'house_owner' column, including NA values
sns.countplot(x='house_owner', data=data_filled, palette='viridis')
plt.title("Distribution of house_owner")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()


# In[13]:


#12) region

#Count missing values in the 'region' column
region_na = data['region'].isna().sum()
print("The na present is ",region_na)


# Provide a summary of the 'region' column (count of each unique value)
region_summary = data['region'].value_counts()
print("Summary of 'region' column:")
print(region_summary)

# 4. Create a bar plot showing the distribution of values in the 'region' column
data_filled = data.copy()
```

```
data_filled['region'] = data_filled['region'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'region' column, including
NA values
sns.countplot(x='region', data=data_filled, palette='viridis')
plt.title("Distribution of region")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()


# In[14]:


#13) fam_income

#Count missing values in the 'fam_income' column
fam_income_na = data['fam_income'].isna().sum()
print("The na present is ",fam_income_na)


# Provide a summary of the 'fam_income' column (count of each unique value)
fam_income_summary = data['fam_income'].value_counts()
print("Summary of 'region' column:")
print(fam_income_summary)

# 4. Create a bar plot showing the distribution of values in the 'fam_income' column
data_filled = data.copy()
data_filled['fam_income'] = data_filled['fam_income'].fillna('Missing')
```

```
# Create a bar plot showing the distribution of values in the 'fam_income' column,
including NA values
sns.countplot(x='fam_income', data=data_filled, palette='viridis')
plt.title("Distribution of fam_income")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()



# In[15]:



#Missing information
missing_info = data.isnull().sum()
print("Missing Information:")
print(missing_info)

# Calculate missing percentage
missing_percentage = 100 * data.isnull().sum() / len(data)
print("Missing Information(%):")
print(missing_percentage)

# Plotting
fig, ax1 = plt.subplots(figsize=(10, 6))

# Actual count of missing values (left y-axis)
color = 'red'
ax1.bar(data.columns, data.isnull().sum(), color=color, label='Count of Missing
Values')
ax1.set_ylabel('Count of Missing Values', color=color)
plt.xticks(rotation=45)
```

```
ax1.tick_params(axis='y', labelcolor=color)

# Percentage of missing values (right y-axis)
ax2 = ax1.twinx()
color = 'black'
ax2.plot(data.columns, missing_percentage, color=color, marker='o',
label='Percentage of Missing Values')
ax2.set_ylabel('Percentage of Missing Values', color=color)
plt.xticks(rotation=45)
ax2.tick_params(axis='y', labelcolor=color)

# Title and labels
plt.title('Bar Chart: Missing Values with Percentage')
plt.xlabel('Variables')
plt.xticks(rotation=45)
plt.legend(loc='upper left')

plt.show()#


# In[16]:



#Addresing data quality issues
#As observed from the data exploartion the data quality issues are as follows and
there are addressed as follows :

#1) Gender
# As observed there are about 1151 'U' obersved in which u is not defined in the
data dictornary will we impute it with male as it is the most occuring mode
```

```
new_data = data.copy()

# Impute 'U' values with 'M' in the 'gender' column of the new DataFrame
new_data['gender'] = new_data['gender'].replace('U', 'M')

# Verify the imputation in the new DataFrame
print(new_data['gender'].value_counts())

# Create a bar plot showing the distribution of values in the 'gender' column
sns.countplot(x='gender', data=new_data, palette='viridis')
plt.title("Distribution of gender")
plt.show()


# In[17]:


#2)Education

#Education has 741 NA valus which will be imputed by mode of the education

# Calculate the mode of the 'education' column
education_mode = new_data['education'].mode()[0]

# Replace NA values in the 'education' column with the mode
new_data['education'] = new_data['education'].fillna(education_mode)

# Verify the imputation by checking if there are any NA values left
print(new_data['education'].isna().sum())
```

```
#plot to verify
data_filled_education = new_data.copy()
data_filled_education['education'] = data_filled_education['education'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'education' column,
including NA values
sns.countplot(x='education', data=data_filled_education, palette='viridis')
plt.title("Distribution of 'education' ")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()


#Count missing values in the 'education' column
edu_na = new_data['education'].isna().sum()
print("The na present is ",edu_na)


# Provide a summary of the 'education' column (count of each unique value)
education_summary = new_data['education'].value_counts()
print("Summary of 'education' column:")
print(education_summary)


# In[18]:


#3) Marriage
#There are NA values which must be replaced by mode
```

```
# Calculate the mode of the 'education' column
Marriage_mode = new_data['marriage'].mode()[0]

# Replace NA values in the 'education' column with the mode
new_data['marriage'] = new_data['marriage'].fillna(Marriage_mode)

# Verify the imputation by checking if there are any NA values left
print(new_data['marriage'].isna().sum())



#plot to verify
data_filled_marriage= new_data.copy()
data_filled_marriage['marriage'] = data_filled_marriage['marriage'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'education' column,
including NA values
sns.countplot(x='marriage', data=data_filled_marriage, palette='viridis')
plt.title("Distribution of 'marriage' ")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()

marriage_summary = new_data['marriage'].value_counts()
print("Summary of 'marriage' column:")
print(marriage_summary)



# In[19]:



#4)Child
```

```python
#there is 0 an undefined entry as they are only 127 entries they can be deleted

# Remove rows where 'child' column has the value '0'
new_data = new_data[new_data['child'] != '0']

# Verify the removal by checking the unique values left in the 'child' column
print(new_data['child'].unique())

data_filled = new_data.copy()
data_filled['child'] = data_filled['child'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'child' column, including
NA values
sns.countplot(x='child', data=data_filled, palette='viridis')
plt.title("Distribution of child")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()

#Count missing values in the 'child' column
child_na = new_data['child'].isna().sum()
print("The na present is ",child_na)



# Provide a summary of the 'child' column (count of each unique value)
child_summary = new_data['child'].value_counts()
print("Summary of 'child' column:")
print(child_summary)
```

```
# In[20]:


#5)house_owner
#there are 3377 NA which must be replaced by mode

house_owner_mode = new_data['house_owner'].mode()[0]


# Replace NA values in the 'education' column with the mode
new_data['house_owner'] = new_data['house_owner'].fillna(house_owner_mode)

# Verify the imputation by checking if there are any NA values left
print(new_data['house_owner'].isna().sum())


#plot to verify
data_filled_house_owner = new_data.copy()
data_filled_house_owner['house_owner'] =
data_filled_house_owner['house_owner'].fillna('Missing')

# Create a bar plot showing the distribution of values in the 'education' column,
including NA values
sns.countplot(x='house_owner', data=data_filled_house_owner, palette='viridis')
plt.title("Distribution of 'house_owner' ")
plt.xticks(rotation=45)  # Optional: Rotate labels if they overlap
plt.show()

house_owner_summary = new_data['house_owner'].value_counts()
print("Summary of 'house_owner' column:")
```

```python
print(house_owner_summary)


# In[21]:


#correlation and bivarient analysis for 5 variables


# In[22]:


#Age vs flag
# 'age' is categorical in this dataset,
plt.figure(figsize=(12, 6))
sns.countplot(x='age', hue='flag', data=new_data, palette='viridis')
plt.title('Purchase Behavior Across Different Age Groups')
plt.xlabel('Age Group')
plt.ylabel('Count')
plt.legend(title='Purchased', loc='upper right')
plt.tight_layout()

# Prepare data for Chi-square test
contingency_table = pd.crosstab(new_data['age'], new_data['flag'])

# Perform Chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Display the test results
print(f'Chi-square Statistic: {chi2}, p-value: {p_value}')
```

```
# Print the results
print(f"Degrees of Freedom: {dof}")
print(f"P-value: {p_value}")
print("Expected Frequencies:", expected)

plt.show()



# In[23]:



#Gender VS Flag (purchased)
purchase_by_gender = new_data.groupby('gender')['flag'].value_counts().unstack()
print(purchase_by_gender)

# Now, for the visual representation
plt.figure(figsize=(8, 6))
sns.countplot(x='gender', hue='flag', data=new_data)
plt.title('Purchase Behavior by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Purchased')
plt.show()

#chisq test
# Prepare data for Chi-square test
contingency_table_gen = pd.crosstab(new_data['gender'], new_data['flag'])

# Perform Chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table_gen)
```

```
# Display the test results
print(f'Chi-square Statistic: {chi2}, p-value: {p_value}')
# Print the results
print(f"Degrees of Freedom: {dof}")
print(f"P-value: {p_value}")
print("Expected Frequencies:", expected)


# In[24]:


#Education vs Flag

plt.figure(figsize=(10, 6))
sns.countplot(x='education', hue='flag', data=new_data, palette='Set2')
plt.title('Purchase Behavior by Education Level')
plt.xlabel('Education Level')
plt.ylabel('Count')
plt.legend(title='Purchased')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Now, for the Chi-square test of independence
contingency_table = pd.crosstab(new_data['education'], new_data['flag'])

# Perform Chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)
```

```
# Display the test results
print(f'Chi-square Statistic: {chi2}, p-value: {p_value}')
# Print the results
print(f"Degrees of Freedom: {dof}")
print(f"P-value: {p_value}")
print("Expected Frequencies:", expected)


# In[25]:



#Marriage vs Flag


plt.figure(figsize=(8, 6))
sns.countplot(x='marriage', hue='flag', data=new_data, palette='coolwarm')
plt.title('Purchase Behavior by Marriage Status')
plt.xlabel('Marriage Status')
plt.ylabel('Count')
plt.legend(title='Purchased')
plt.show()

# Now, for the Chi-square test of independence
contingency_table = pd.crosstab(new_data['marriage'], new_data['flag'])

# Perform Chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Display the test results
print(f'Chi-square Statistic: {chi2}, p-value: {p_value}')
# Print the results
```

```python
print(f"Degrees of Freedom: {dof}")
print(f"P-value: {p_value}")
print("Expected Frequencies:", expected)


# In[26]:


#occupation vs Flag

# Visual analysis for 'occupation'
plt.figure(figsize=(10, 6))
sns.countplot(x='occupation', hue='flag', data=new_data, palette='viridis')
plt.title('Purchase Behavior by Occupation')
plt.xlabel('Occupation')
plt.ylabel('Count')
plt.legend(title='Purchased')
plt.xticks(rotation=45)
plt.show()

# Now, for the Chi-square test of independence
contingency_table = pd.crosstab(new_data['occupation'], new_data['flag'])

# Perform Chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Display the test results
print(f'Chi-square Statistic: {chi2}, p-value: {p_value}')
# Print the results
print(f"Degrees of Freedom: {dof}")
```

```python
print(f"P-value: {p_value}")
print("Expected Frequencies:", expected)


# In[ ]:


#Converting all character as factors

# Loop through each column and convert object type columns to categorical
for column in new_data.columns:
    if new_data[column].dtype == 'object':
        new_data[column] = new_data[column].astype('category')

# Verify the conversion by printing the data types of the DataFrame columns
print(new_data.dtypes)


# In[27]:


# Set random seed for reproducibility
np.random.seed(40412492)


# In[28]:


from sklearn.model_selection import train_test_split
```

```
#splitting data into test and train
X = new_data.drop('flag', axis=1)
y = new_data['flag']

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=40412492)



# In[29]:



#Scaling the data


from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

# Identifying categorical and numerical columns
categorical_cols = X.select_dtypes(include=['object', 'bool']).columns
numerical_cols = X.select_dtypes(include=['int64', 'float64']).columns

# Define transformers for the preprocessing step
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(), categorical_cols)
    ])
```

```python
# Fit the preprocessor and transform the training and testing data
X_train_preprocessed = preprocessor.fit_transform(X_train)
X_test_preprocessed = preprocessor.transform(X_test)


# In[30]:


#import necessary lib
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, label_binarize
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
roc_auc_score


# In[33]:


#Model 1
# Logistic Regression model with hyperparameters
logistic_regression_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
```

```
    ('classifier', LogisticRegression(C=1.0, max_iter=10000,
random_state=40412492))
])

# Train the model
logistic_regression_model.fit(X_train, y_train)

# Predict on the testing data
y_pred = logistic_regression_model.predict(X_test)
y_proba = logistic_regression_model.predict_proba(X_test)[:, 1]

# Calculate metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy:.4f}\nPrecision: {precision:.4f}\nRecall: {recall:.4f}\nF1
Score: {f1:.4f}')

# Plotting Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

# Adjusting for the ValueError in the roc_curve function
from sklearn.metrics import roc_curve, auc
```

```python
# Calculate the ROC curve data
fpr, tpr, thresholds = roc_curve(y_test, y_proba, pos_label='Y')
roc_auc = auc(fpr, tpr)

# Plotting the ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()



# In[44]:



#Model 2
from sklearn.ensemble import RandomForestClassifier

# Define the pipeline for the Random Forest model
rf_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=40412492))
])
```

```
# Setup the grid search for the Random Forest model
param_grid_rf = {
    'classifier__n_estimators': [100, 200],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 5]
}

grid_search_rf = GridSearchCV(rf_pipeline, param_grid_rf, cv=5, scoring='accuracy',
verbose=1)

#Grid search
# Perform the grid search
grid_search_rf.fit(X_train, y_train)

# Extract the best estimator
best_rf = grid_search_rf.best_estimator_

#Run model
# Predict on the testing data using the best model
y_pred_best_rf = best_rf.predict(X_test)

# Since RandomForest does not have predict_proba method by default, check
before calling it
y_proba_best_rf = best_rf.predict_proba(X_test)[:, 1] if hasattr(best_rf,
'predict_proba') else None

# Calculate metrics
accuracy_best_rf = accuracy_score(y_test, y_pred_best_rf)
precision_best_rf = precision_score(y_test, y_pred_best_rf, average='weighted',
zero_division=0)
```

```
recall_best_rf = recall_score(y_test, y_pred_best_rf, average='weighted')
f1_best_rf = f1_score(y_test, y_pred_best_rf, average='weighted')

print(f'Best Random Forest Metrics:\nAccuracy: {accuracy_best_rf:.4f}\nPrecision:
{precision_best_rf:.4f}')
print(f'Recall: {recall_best_rf:.4f}\nF1 Score: {f1_best_rf:.4f}')

#plot cf and roc

cm_best_rf = confusion_matrix(y_test, y_pred_best_rf)
sns.heatmap(cm_best_rf, annot=True, fmt="d", cmap="Greens")
plt.title('Confusion Matrix for Best Random Forest')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

if y_proba_best_rf is not None:
    fpr_best_rf, tpr_best_rf, _ = roc_curve(y_test, y_proba_best_rf, pos_label='Y')
    roc_auc_best_rf = auc(fpr_best_rf, tpr_best_rf)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr_best_rf, tpr_best_rf, color='darkorange', lw=2, label=f'Best Random
Forest ROC curve (area = {roc_auc_best_rf:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve for Best Random Forest')
    plt.legend(loc="lower right")
```

```
plt.show()
```