Artificial Intelligence

COSTUMER_INSURANCE_LTV.csv Dataset

Members:
Israa Atike (D00262160)
Yu Kang Ong (D00262134)
Kaylee Jacqueline Wijaya (D00262128)

Table of Contents

Table of Contents 1
Introduction 2
Problem Statement 3
Roadmap 4
1. Dataset selection: 4
2. Data cleaning: 6
3. Exploratory data analysis: 7
4. Feature Engineering: 22

Artificial Intelligence Technologies 29
Solution Evaluation 30
Conclusion 32
References 33
Appendix 34

Introduction

Artificial intelligence has rapidly emerged as a strong force in diverse and different domains, offering easier solutions to real-world complicated projects, Artificial Intelligence(AI) has been a big part of our daily lives for the past 3 years since the launch of ChatGPT, including machine learning, deep learning and generative Ai tools, therefore addressing it and trying to comprehend its algorithms and strategies, makes it more understandable as well as more enjoyable, not only but also it enables us to use it the right way and in a more effective way.

In this report, we will explain our application of Artificial Intelligence tools and techniques to our chosen dataset called CUSTOMER_INSURANCE_LTV.csv.

In this report, we will demonstrate all steps of our AI project including, dataset selection, data cleaning, exploratory data analysis, feature engineering and finally Machine Learning, as well as our problem statement, roadmap and the solution we suggest, the most crucial aim of this project is to identify and recommend the most effective AI algorithm to this dataset.

Problem Statement

In this project, we are trying to predict whether a customer will buy insurance based on different attributes.

Our research question is: **What factors influence insurance purchase the most, and which Machine Learning or AI algorithms provide the most accurate prediction of our response variable?**

AI is a really good tool for answering the research question, because of the following reasons:

- It can process large datasets efficiently
- It usually results in a highly accurate mathematical model
- It can detect complex and non-linear relationships between multiple variables

The following are the steps taken to approach the problem.

1. Load the data
2. Determine the dependent variable and independent variables of the dataset
3. Perform data cleaning
4. Exploratory data analysis
5. Feature engineering
6. Remove multicollinearity and reduce outliers
7. Train/Test/Validation Split (60/20/20)
8. Choosing the model
9. Model training
10. Testing the model
11. Evaluate model

Roadmap

1. Dataset selection:

The dataset chosen for this project is called CUSTOMER_INSURANCE_LTVA.csv we got this dataset from GitHub please check the link below,
https://github.com/AnalyticsandDataOracleUserCommunity/MachineLearning/blob/master/cust_insur_ltv/CUSTOMER_INSURANCE_LTV.csv
We chose a dataset that is related to insurance since, it has real-world relevance, which enables us to have a practical application on the dataset, furthermore it is rich and diverse in features since it provides different types of information such as demographic, social and financial data, not only but also because it is perfect for classification algorithm in Machine learning, and finally because it allows us to discover new insights that will help insurance companies in their predictions of the number of people who will choose them as an insurance provider.
The dataset we chose contains 31 columns and 13880 rows representing customer attributes and information from various regions of the United States of America, the features of this dataset include both categorical and numerical variables related to personal, financial, social, demographic and behavioural factors that influence the response variable/ dependent variable which is 'BUY_INSURANCE'.

A- Personal info attributes:

'CUSTOMER_ID': which specifies the ID of each customer.
'FIRST_NAME': which specifies the first name of the customer.
'LAST_NAME': which specifies the last name of the customer.

b- Social attributes:

'MARITIAL_STATUS': which specifies the customer's marital status which is 'married', 'divorced', 'widowed', 'single' and 'others'.
'GENDER': which specifies the customer's gender which is 'M' for Male or 'F' for Female.
'HAS_CHILDREN' specifies if the customer has children or not, which is either '0' for no children or '1' for having children.
'NUM_DEPENDENTS': which specifies the number of people that are dependent on the customer which ranges from '0' to '6' people.
'AGE': which specifies the customer's age which ranges from '17' years old to '84' years old.
'CAR_OWNERSHIP': which specifies if the customer owns a car or not '0' indicates the non-ownership of a car and '1' indicates the ownership of a car.
'HOME_OWNERSHIP': which specifies if the customer owns a house or not, '0' for not owning a house, '1' for owning one house, or '2' for owning 2 houses.

c- Demographic attributes:

‘REGION’: which specifies the region where the customer resides, which is either ‘Midwest’, ‘Northeast’, or ‘South’. ‘Southwest’ or ‘West’.
‘STATE’: which specifies the states where the customer resides and which is a long list of USA states.

d- Behavioral attributes:

‘CUSTOMER_TENURE’: which specifies the number of years the customer has been involved with the insurance provider and it ranges from ‘1’ year to ‘5’ years.
‘PROFESSION’: which specifies the job of the customer which is a long list of jobs.
‘NUM_ONLINE_TRANS’: which specifies the number of online transactions performed by the customer, which is also a long list of integers.
‘NUM_TRANS_KIOSK’: which specifies the number of transactions completed by the customer at self-service kiosks which ranges from ‘0’ to ‘10’ transactions.
‘NUM_TRANS_TELLER’: which specifies the number of transactions performed by a customer with a human bank teller which ranges from ‘0’ to ‘9’ transactions.
‘NUM_TRANS_ATM’: which specifies the transactions performed by a customer in an ATM which ranges from ‘0’ to ‘8’ transactions.
‘MONTHLY_CHECKS’: which specifies the number of checks written or processed by a customer every month which ranges from ‘0’ to ‘18’ transactions.

f- Financial attributes:

‘CREDIT_BALANCE’: which specifies the credit balance of the customer which is a numerical variable.
‘MORTGAGE_AMOUNT’: which specifies the amount of mortgage held by a customer which is a numerical variable.
‘BANK_FUNDS’: which specifies the total funds available in a customer account which is a numerical variable.
‘INCOME’: which specifies the income of the customer which is a numerical variable.
‘CREDIT_CARD_LIMITS’: which specifies the limits on the customer credit card which is a numerical variable.
‘MONEY_MONTHLY_OVERDRAWN’: which specifies the average amount of money the customer overdraws on their account per month which is a numerical variable.
‘LTV’: specifies the loan-to-value of the customer which is a numerical variable.
‘TOTAL_AUTOM_PAYMENTS’: which specifies the total monetary value of automated payments made by a customer which is a numerical variable.
‘CHECKING_BALANCE’: which specifies the current balance in a customer’s checking account which is a numerical variable.
‘LTV_BIN’: which specifies the categorical classification of a customer’s loan-to-value ratio, which is either ‘Low’, ‘Medium’, ‘High’ or ‘Very High’.
‘NUM_MORTGAGES’: which specifies the number of mortgages held by a customer which is either ‘0’, ‘1’ or ‘2’.

‘BUY_INSURANCE’: this is our response variable which specifies if that customer buys or does not buy the insurance ‘Yes’ for buying the insurance and ‘No’ for not buying it.

As mentioned above my **dependent variable** is **‘BUY_INSURANCE’** which is the same variable that we will be predicting using Machine Learning later on.
And **all** the other **variables** are considered as **independent variables**.

2. Data cleaning:

Although the dataset has no null values, there is still some data cleaning to do.

First, we remove a few irrelevant columns that do not affect whether a customer will buy insurance or not, which are the customer’s ID, customer’s first name and customer’s last name.

Next, if we look at the column `PROFESSION`, we will notice there are too many unique values. We try to reduce the unique values by grouping similar professions with their respective field. In the end, we ended up with 12 unique values.

However, there are around 3300 rows of data with undefined professions. To resolve this, we shall replace them with the most frequent profession, which in this case is IT. This is also known as categorical imputer. The 2 figures below show the count of professions before the categorical imputer (Figure 1a) and after the categorical imputer (Figure 1b).

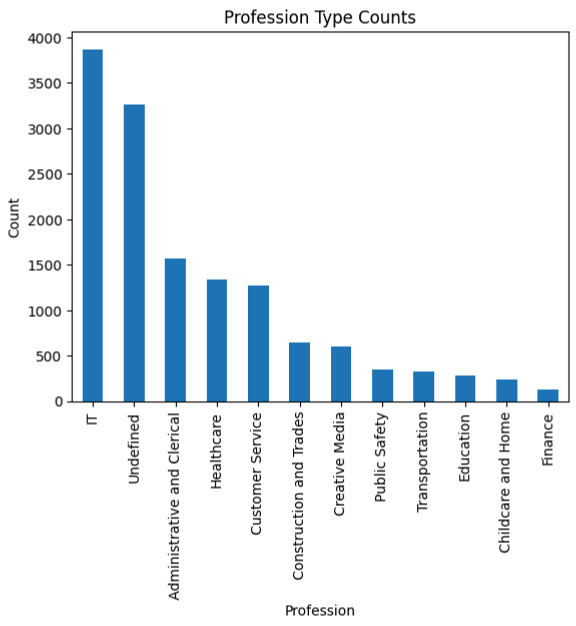


Figure 1a: Before categorical imputer

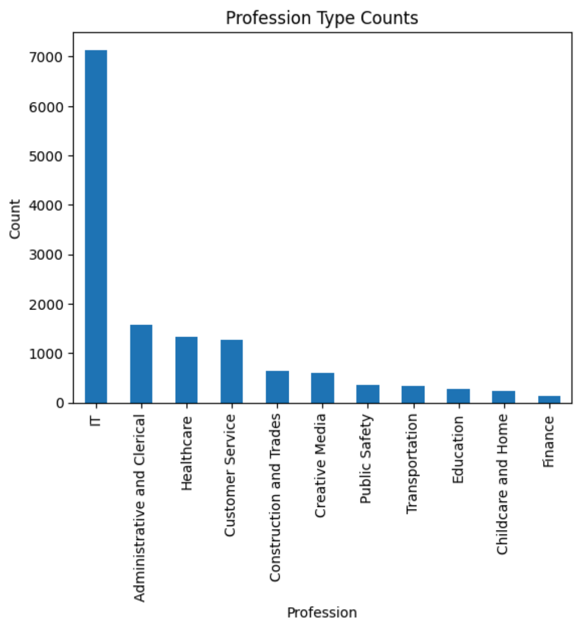


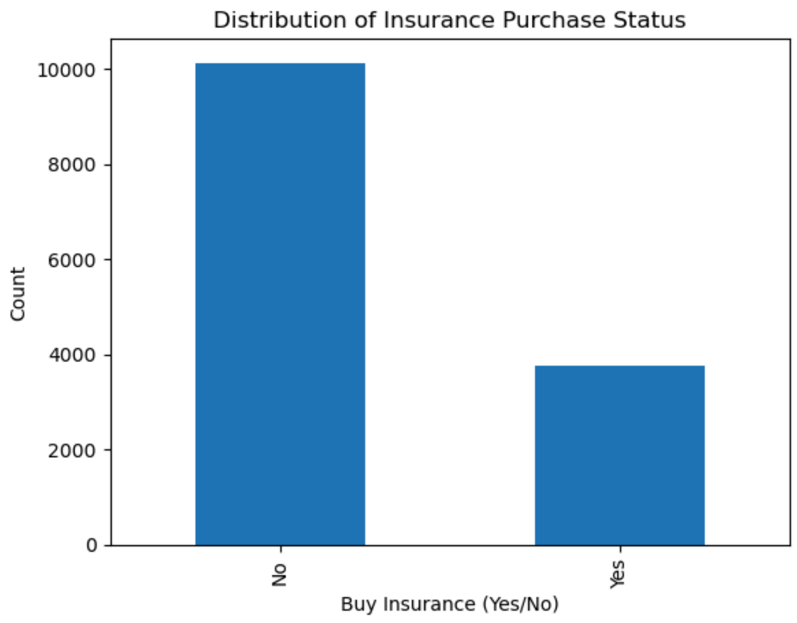
Figure 1b: After categorical imputer

Also, we have changed the state name from an abbreviation to full name. [1]
Note that this is just to know the states when doing exploratory data analysis, and it does not affect the result of the model.

3. Exploratory data analysis:

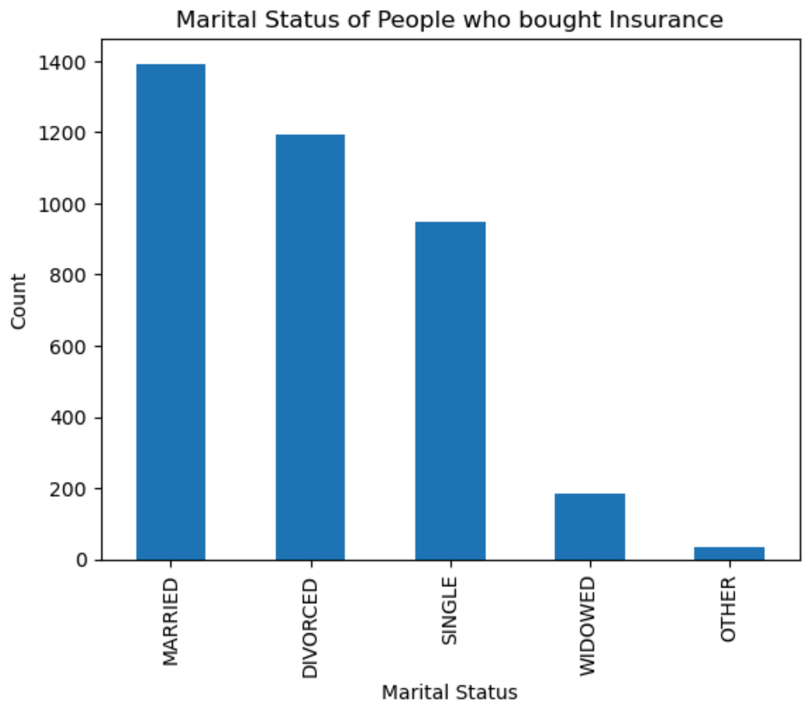
In this section, we will be including all the visualisations and the graphs that we produced to explore the dataset. The code for this section will be provided in the Appendix, since it is simple we will not be explaining it.

'BUY_INSURANCE' bar chart:



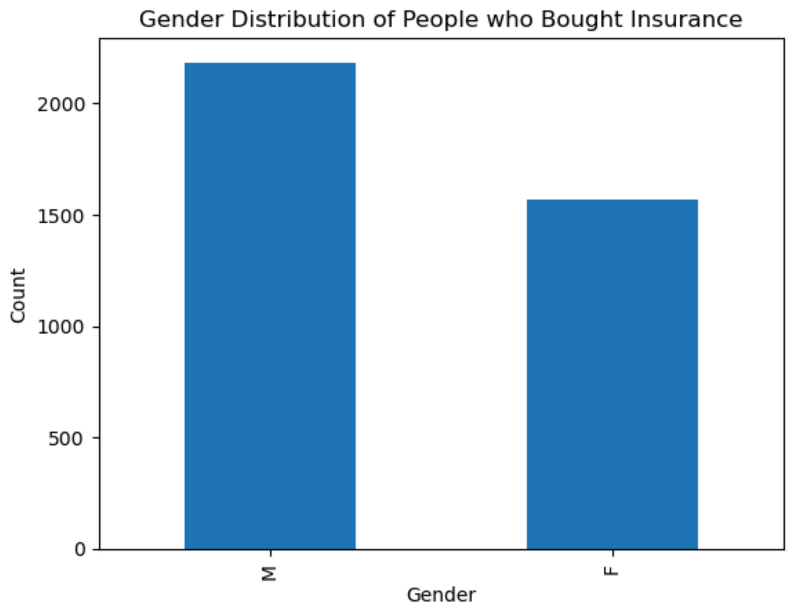
The above bar chart illustrates the distribution of insurance purchases and we can see that the majority of people did not purchase the insurance while a smaller portion of people did purchase it.

‘MARITIAL_STATUS’ bar chart:



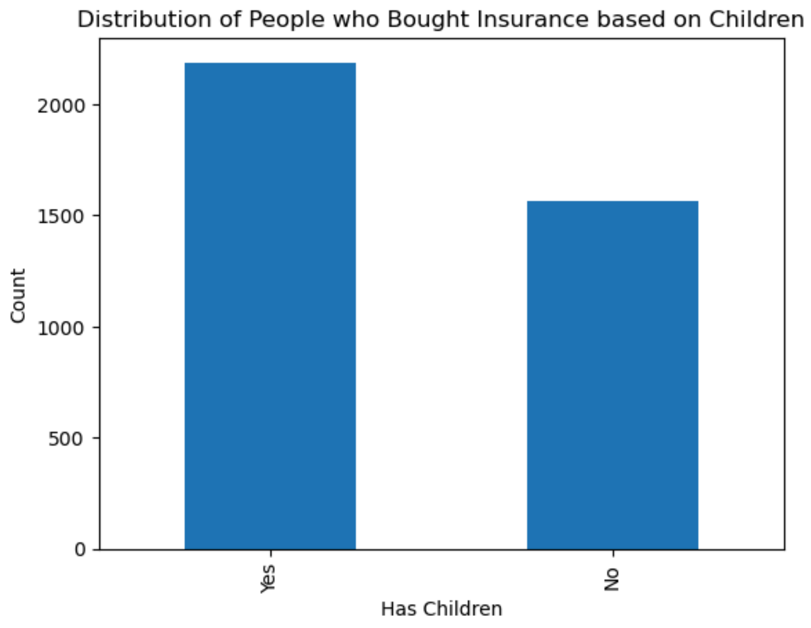
The above bar chart illustrates the distribution of the marital status of customers of the insurance, we can see that most of them are married and then become divorced in second place and then single in third place, widowed in fourth place and finally others in the last place.

‘GENDER’ bar chart:



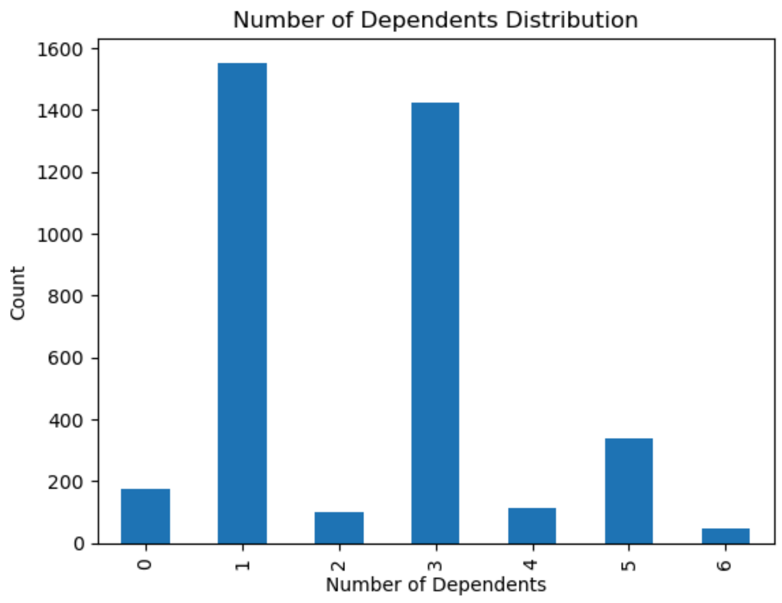
The above bar chart illustrates the distribution of the gender of the customers of the insurance, we can see that most of them are men with more than 2000 customers and then females with almost 1500 customers.

‘HAS_CHILDREN’ bar chart:



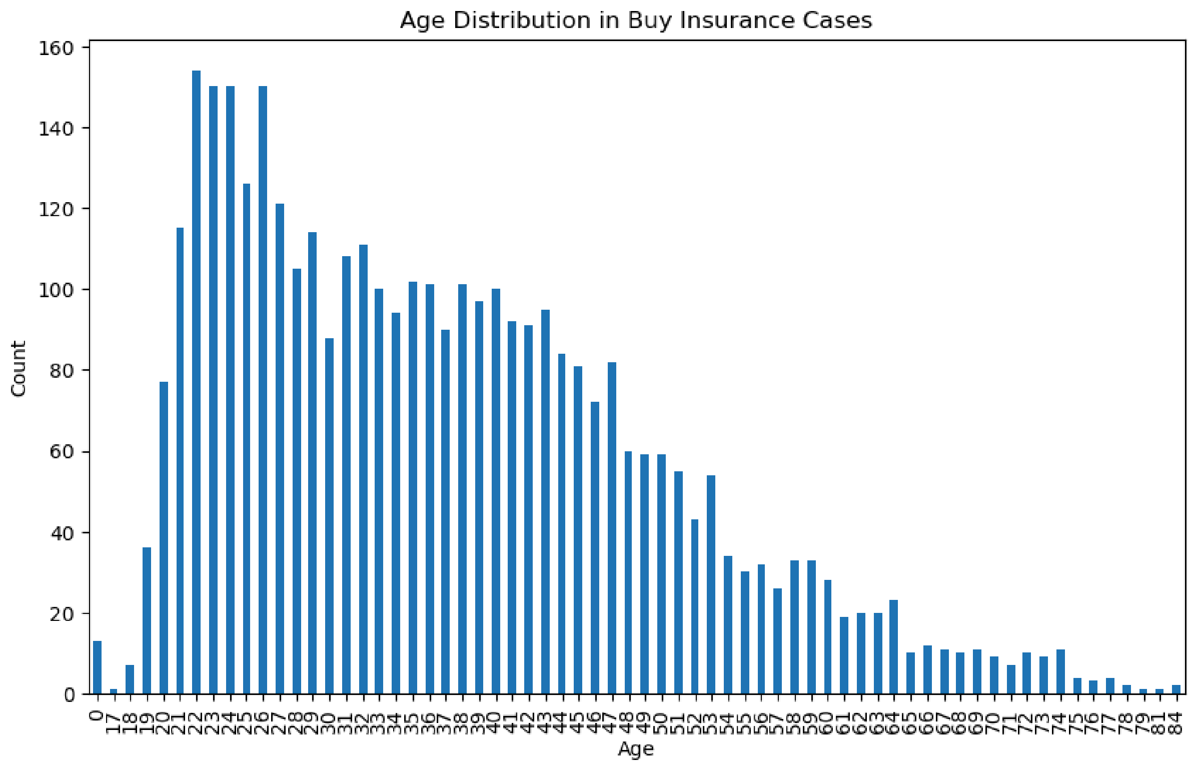
The above bar chart illustrates the distribution of the customers of the insurance if either they have children or not, we can see that the majority do have kids with almost 2200 customers, and fewer customers do not have kids with almost 1500 customers.

‘NUM_DEPENDENTS’ bar chart:



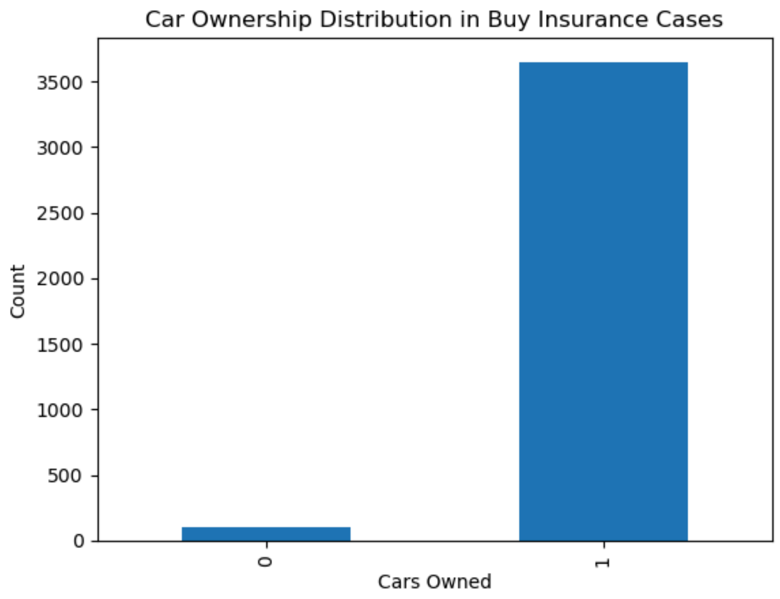
The above bar chart illustrates the distribution of the number of dependents of the customers of the insurance, we can see that most customers have one dependent or 3 dependents, while a minority have 0, 2, 4, 5 or 6 dependents.

‘AGE’ bar chart:



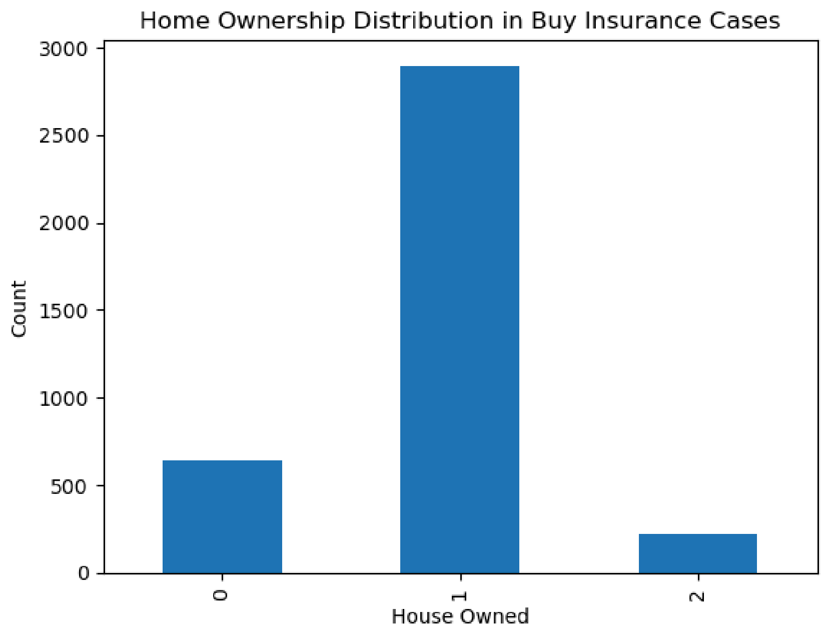
The above bar chart illustrates the distribution of the age of the customers of the insurance, we can see that it ranges from 17 years old to 84 years old with a peak for people who are in their twenties and early thirties.

‘CAR_OWNERSHIP’ bar chart:



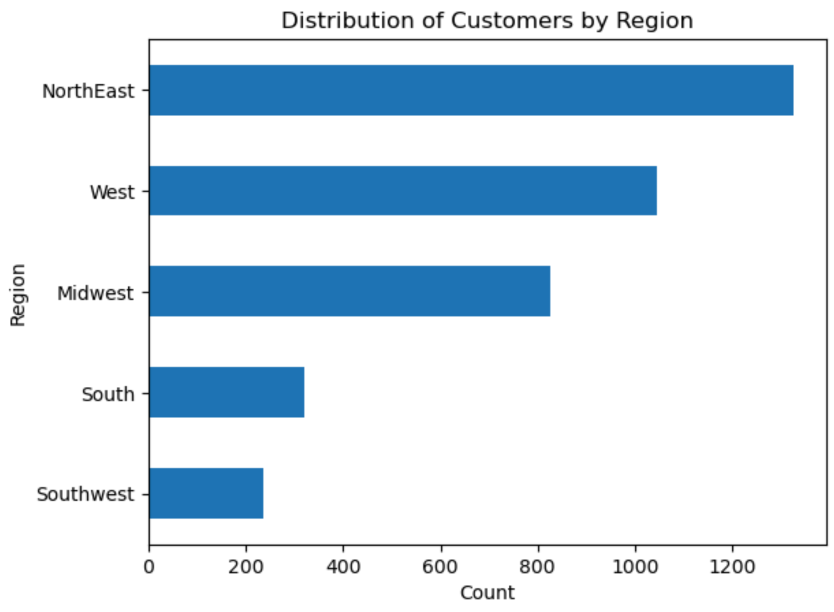
The above bar chart illustrates the distribution of the customers of the insurance of whether they own a car or not, we can see that the majority does own a car with more than 3500 customers, and with a minority of not owning a car with less than 200 customers.

‘HOME_OWNERSHIP’ bar chart:



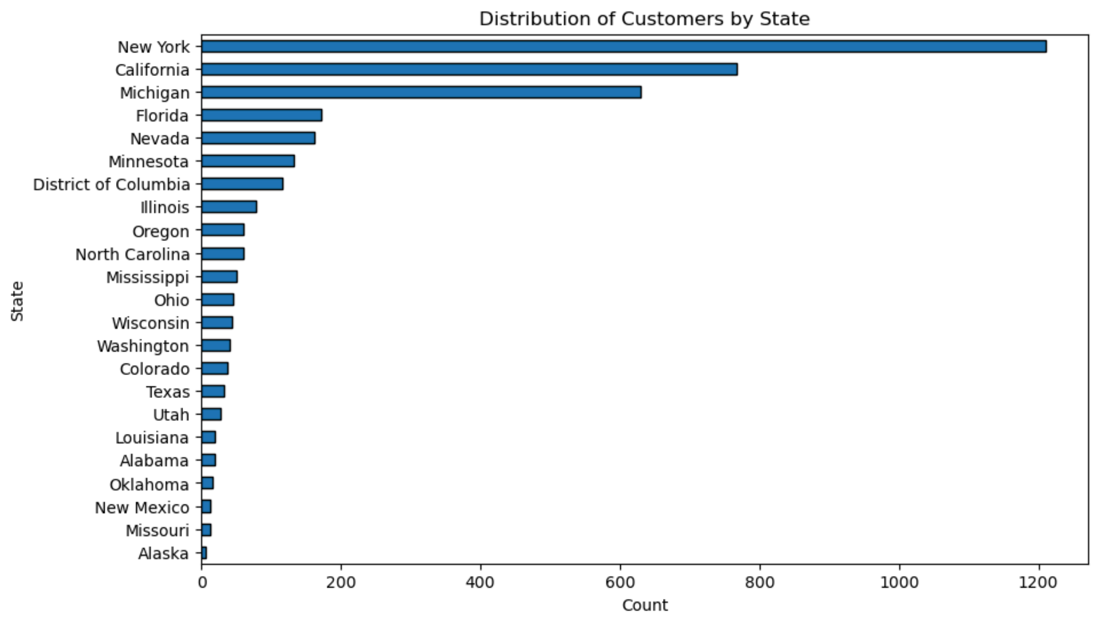
The above bar chart illustrates the distribution of the customers of the insurance of whether they own a home or not, we can see that the majority of customers own one house almost 3000 customers, and then those not owning a home come second with more than 500 customers, and finally owning two homes comes last with less than 300 customers.

‘REGION’ horizontal bar chart:



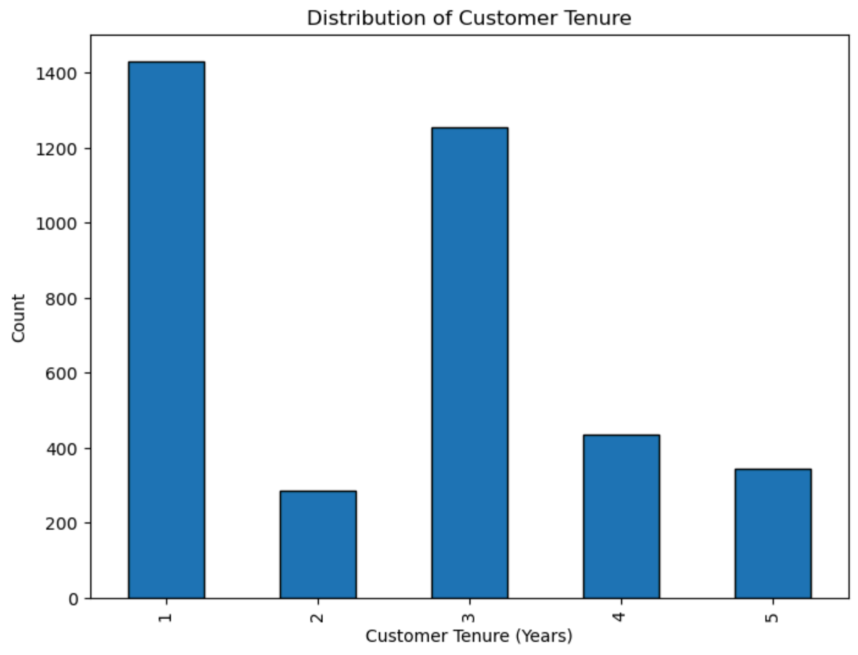
The above bar chart illustrates the distribution of the region of the customers of the insurance, we can see that most of them reside in the NorthEast and respectively other regions as well.

‘STATE’ horizontal bar chart:



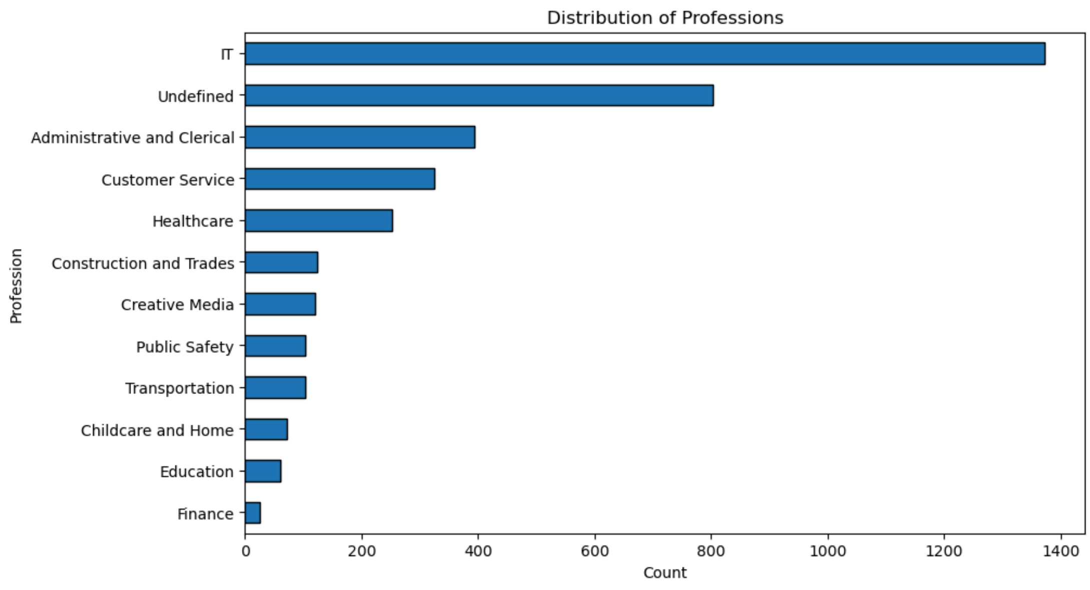
The above bar chart illustrates the distribution of the state of the customers of the insurance, we can see that most of them reside in New York, California and Michigan.

'CUSTOMER_TENURE' bar chart:



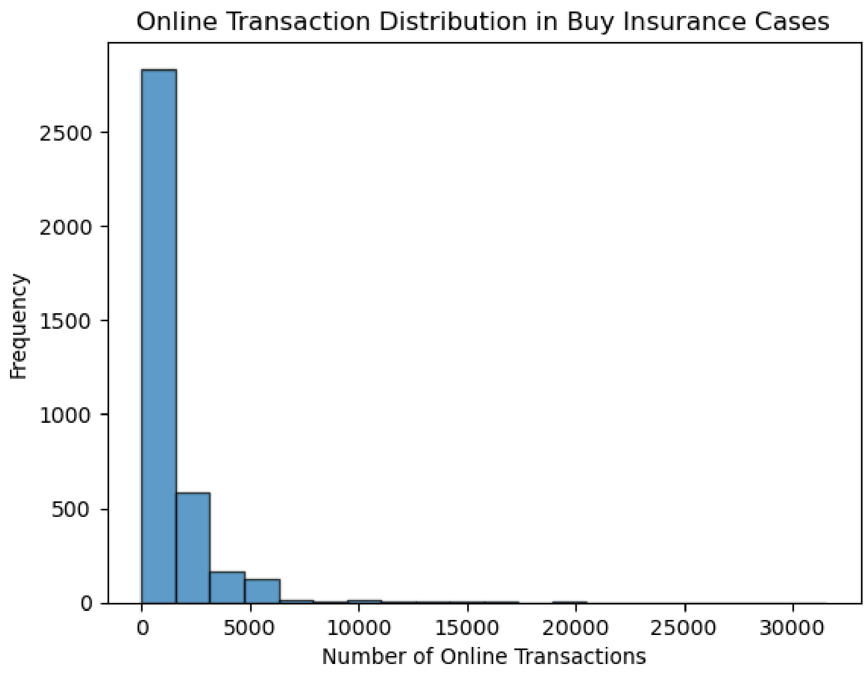
The above bar chart illustrates the distribution of the customer tenure of the customers of the insurance, we can see that most of them have been with the insurance providers for 1 or 3 years and a minority have been for 2, 4 and 5 years.

‘PROFESSION’ bar chart:



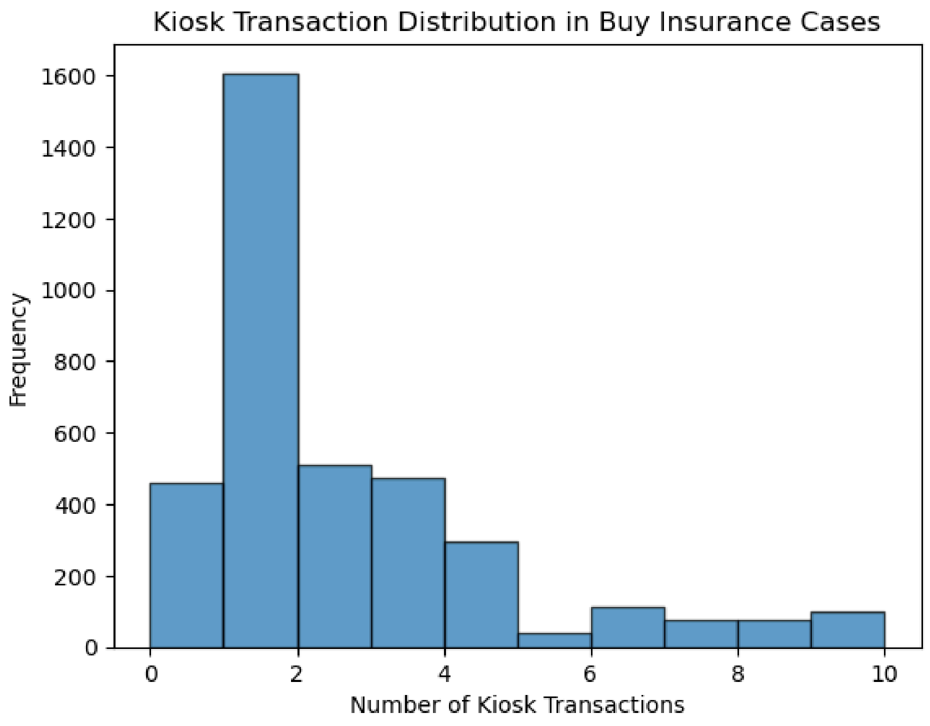
The above bar chart illustrates the distribution of the profession of the customers of the insurance, and we can see that most of them work in IT.

‘NUM_ONLINE_TRANS’ histogram:



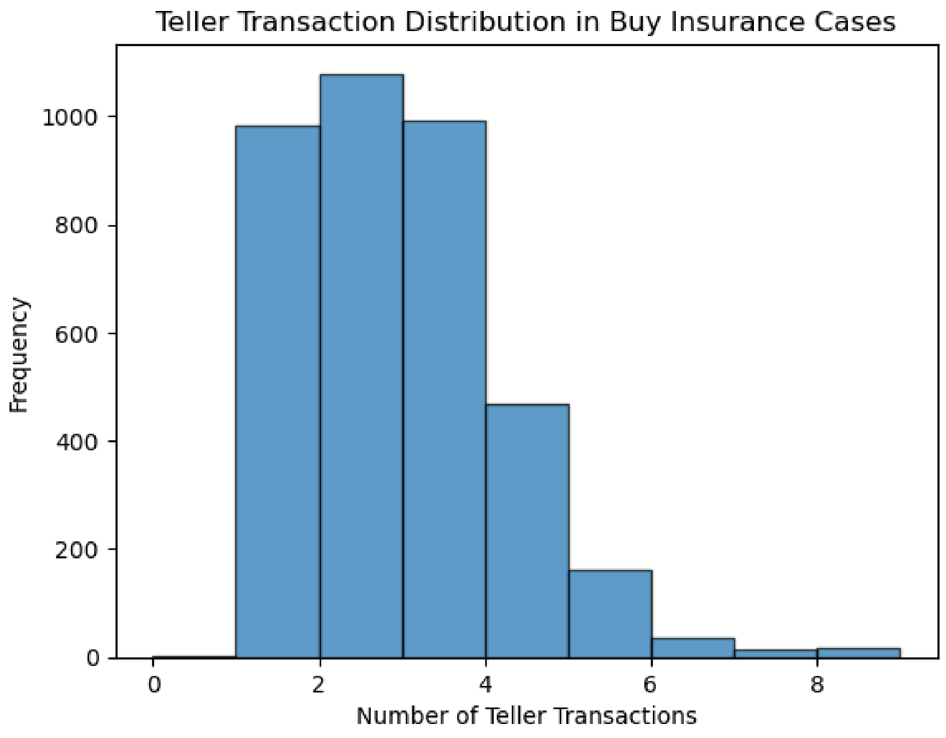
The above histogram illustrates the distribution of the number of online transactions of the customers of the insurance, we can see that the distribution is very skewed to the left and that the majority has less than 2000 numbers of online transactions.

‘NUM_TRANS_KIOSK’ histogram:



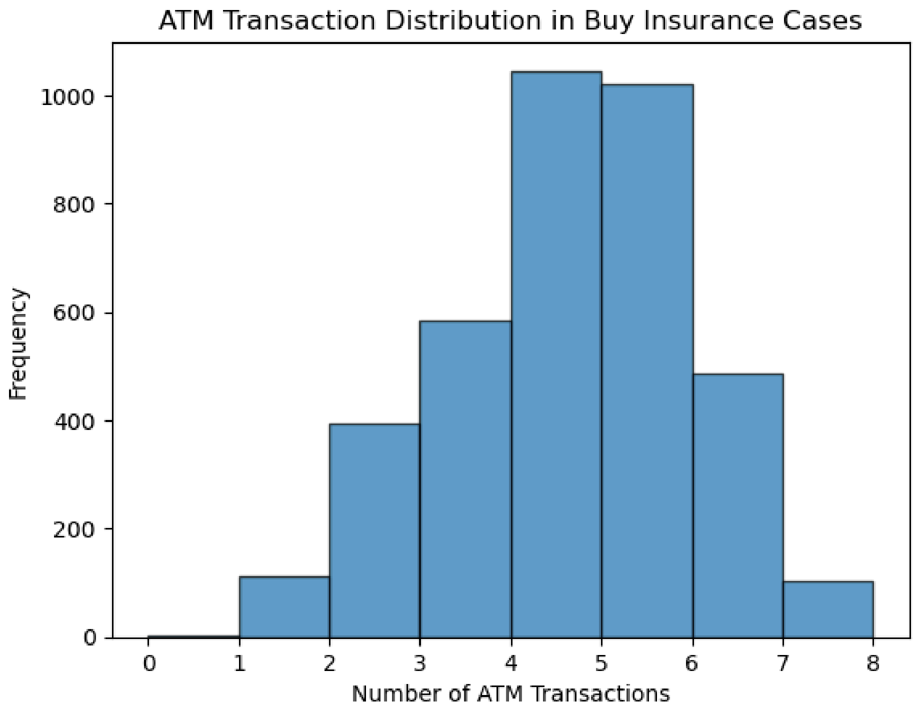
The above histogram illustrates the distribution of the number of kiosk transactions of the customers of the insurance, we can see that most of them did not have many kiosk transactions which is why the distribution is skewed to the left.

‘NUM_TRANS_TELLER’ histogram:



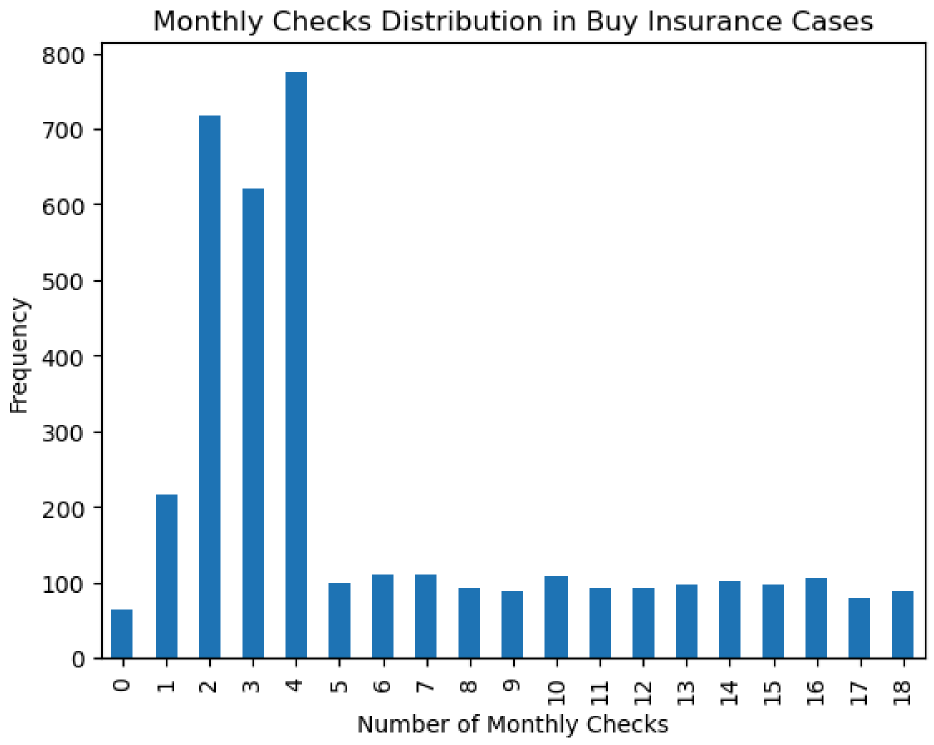
The above histogram illustrates the distribution of the number of teller transactions of the customers of the insurance, we can see that most of them are between 1 and 4 which is why the distribution is skewed to the left, with only very few transactions that are greater than 6.

‘NUM_TRANS_ATM’ histogram:



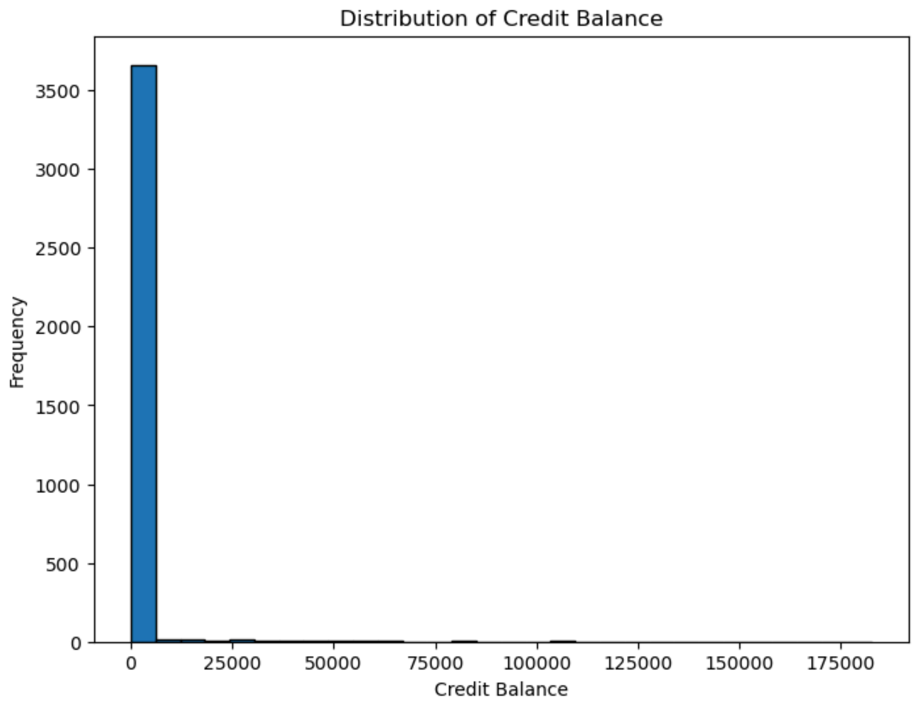
The above histogram illustrates the distribution of the number of transactions using an ATM of the customers of the insurance, the distribution looks more or less symmetrical with a peak between 4 and 6 transactions.

‘MONTHLY_CHECKS’ bar chart:



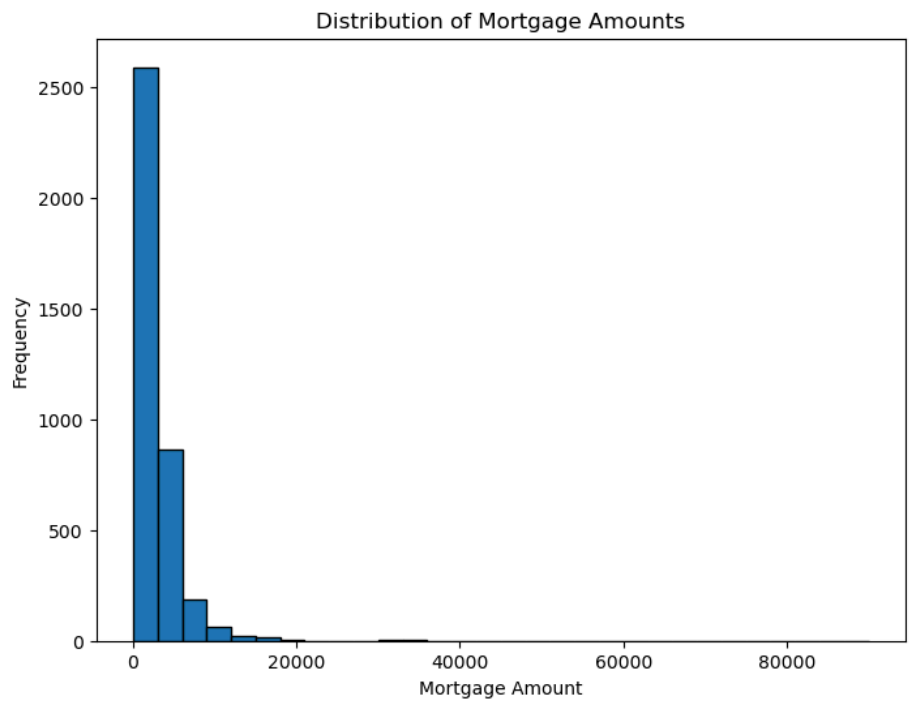
The above bar chart illustrates the distribution of the monthly checks of the customers of the insurance, we can see that 4, 3 and 2 are the numbers with the highest frequencies.

‘CREDIT_BALANCE’ histogram:



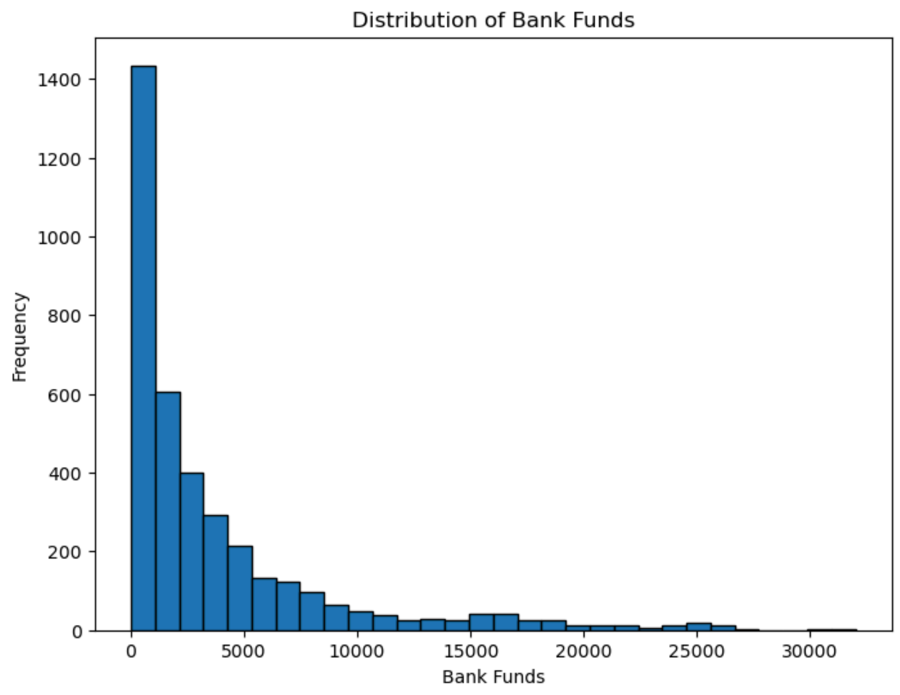
The above histogram illustrates the distribution of the credit balance of the customers of the insurance, we can see that almost all customers have less than 25000 dollars in their balances.

‘MORTGAGE_AMOUNT’ histogram:



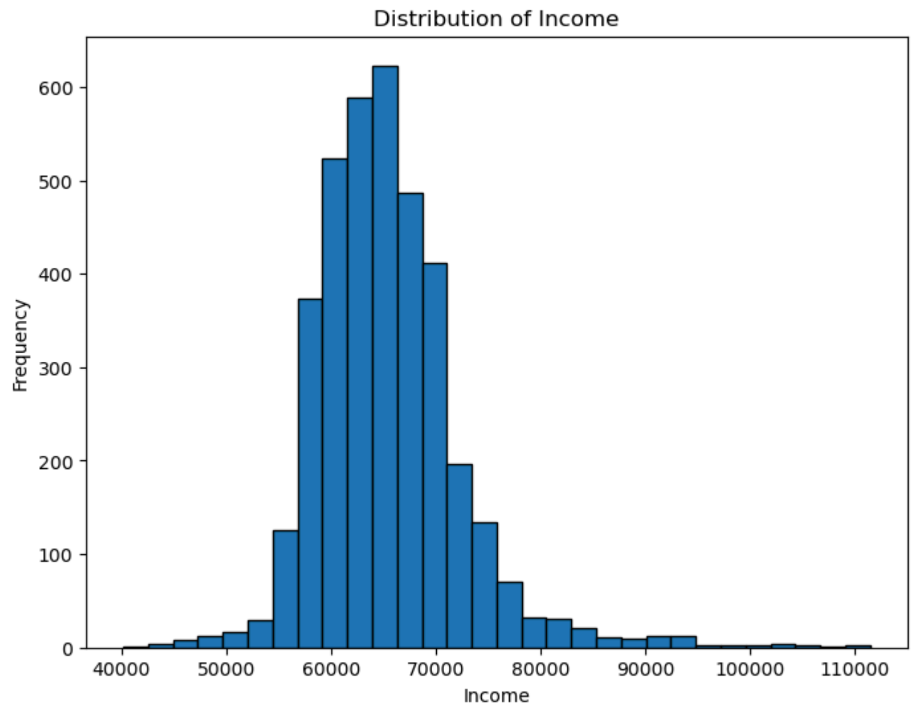
The above histogram illustrates the distribution of the mortgage amounts of the customers of the insurance, we can see that most of them have less than 20000 dollars of mortgage, with a very skewed to the left distribution.

‘BANK_FUNDS’ histogram:



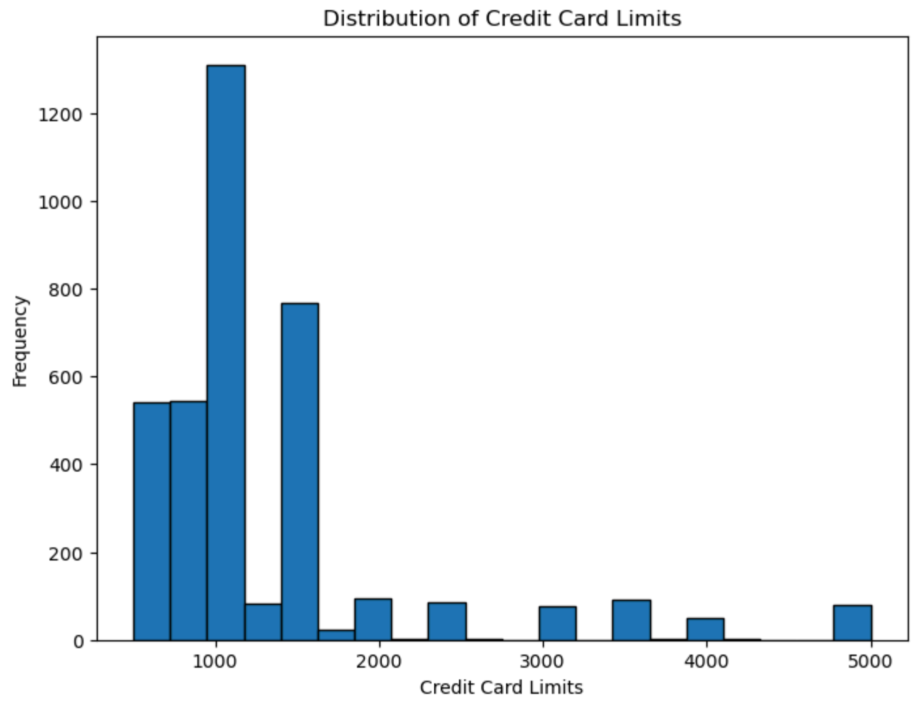
The above histogram illustrates the distribution of the bank funds of the customers of the insurance with a skewed to the left distribution since most customers have less than 5000 dollars in their bank funds.

‘INCOME’ histogram:



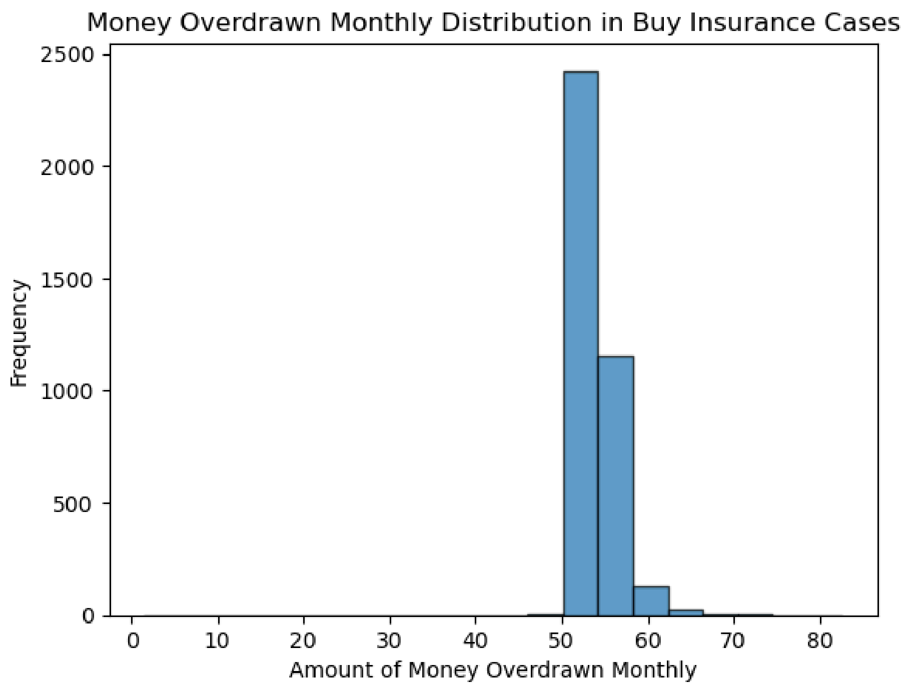
The above histogram illustrates the distribution of the income of the customers of the insurance, we can see that the distribution is very symmetric, since when it comes to incomes most of the insurance customers are paid more or less the same income.

‘CREDIT_CARD_LIMITS’ histogram:



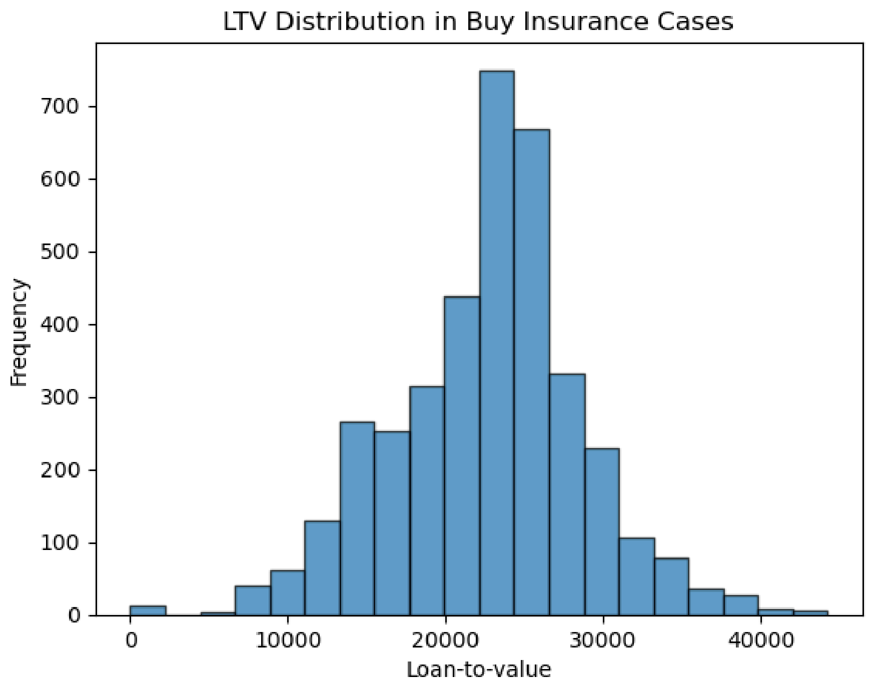
The above histogram illustrates the distribution of the credit card limits of the customers of the insurance, we can see that the distribution is skewed to the left and that most credit card limits are less than 1700 dollars.

‘MONEY_MONTLY_OVERDRAWN’ histogram:



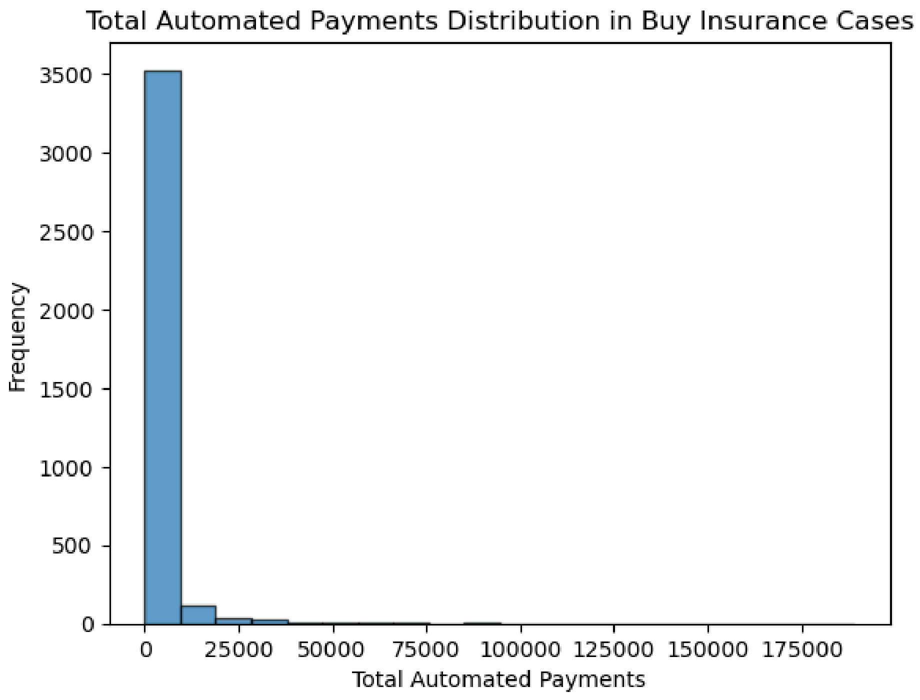
The above histogram illustrates the distribution of the money monthly overdrawn by the customers of the insurance, we can see that most of them are between 50 and 58.

‘LTV’ histogram:



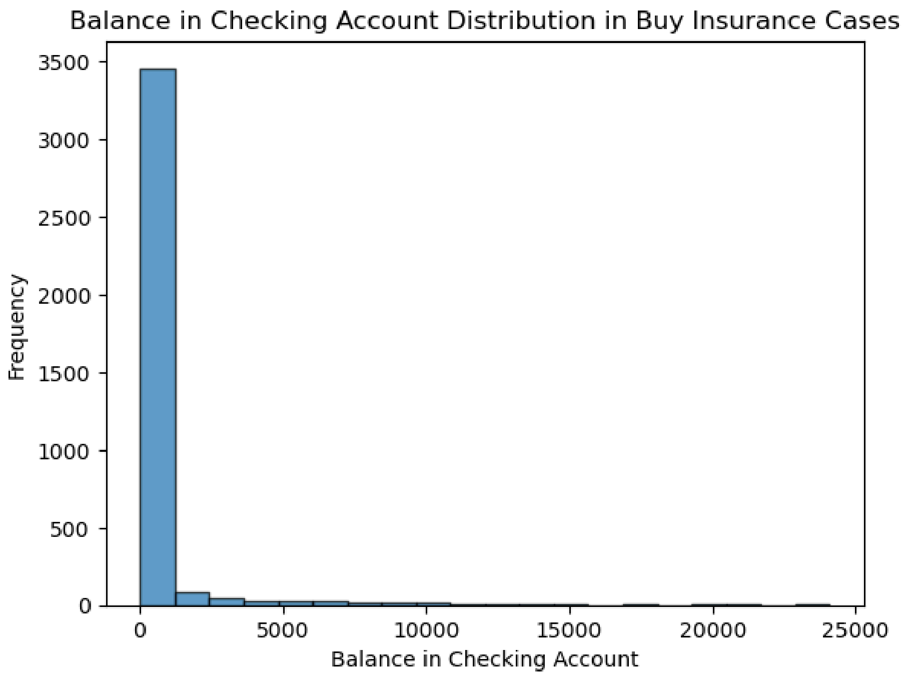
The above histogram illustrates the distribution of the LTV of the customers of the insurance, we can see that the distribution is very symmetric, with a peak between 22000 and 26000.

‘TOTAL_AUTOM_PAYMENTS’ histogram:



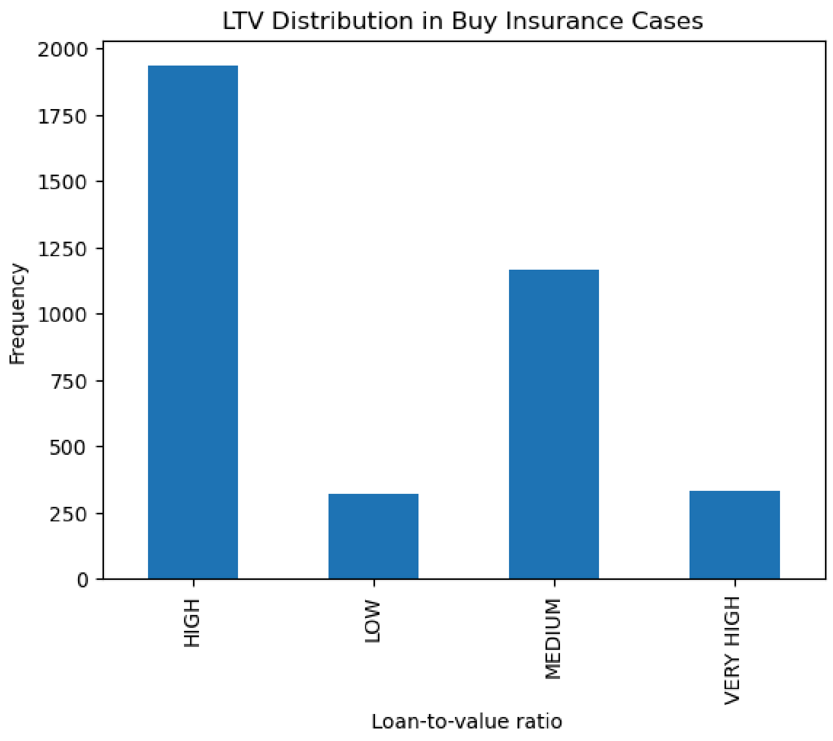
The above histogram illustrates the distribution of the total auto payments of the customers of the insurance, we can see that most of them are less than 25000 dollars which makes the distribution very skewed to the left.

‘CHECKING_BALANCE’ histogram:



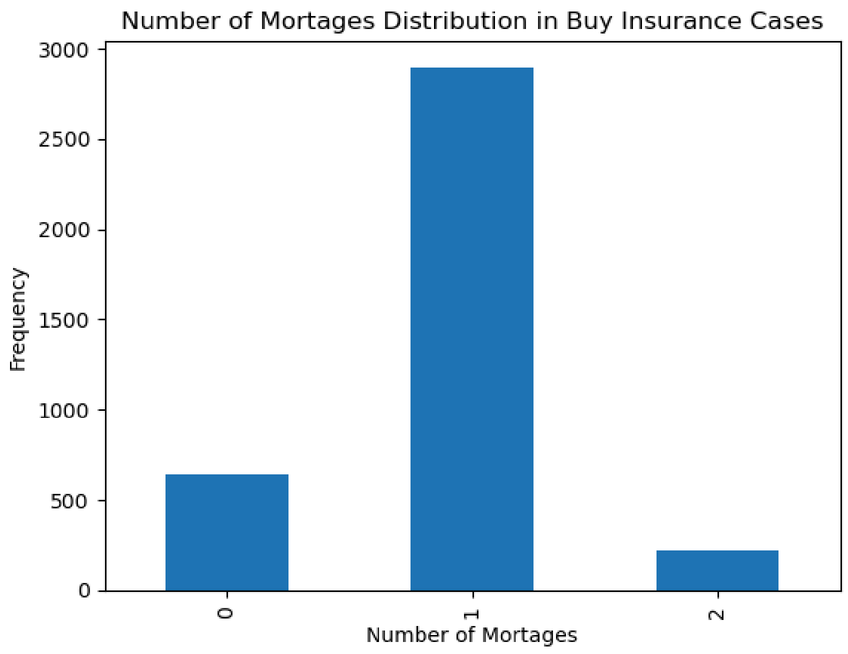
The above histogram illustrates the distribution of the checking balance of the customers of the insurance that is very skewed to the left since most of the customers have a balance that is less than 5000 dollars.

‘LTV_BIN’ bar chart:



The above bar chart illustrates the distribution of the LTV bin of the customers of the insurance, we can see that most of them have a high loan-to-value ratio with a low LTV in second place, a medium one for the 3rd place and finally a very high one for the last place.

‘NUM_MORTGAGES’ bar chart:



The above bar chart illustrates the distribution of the number of mortgages of the customers of the insurance. we can see that most of them have 1 mortgage and other customers have no mortgages for the second place and finally some customers have 2 mortgages for the last place.

4. Feature Engineering:

For feature engineering, the first thing we did was convert categorical features to numeric ones.

- For the `GENDER` column, encode male as 0, and female as 1.
- For the `STATE`, `REGION` and `PROFESSION` columns, use a one-hot encoding.
- For the `LTV_BIN` column, encode low as 0, medium as 1, high as 2, very high as 3.
- For the `MARITAL_STATUS` column, encode single as 0, married as 1, divorced, widowed and other as 2.
- For the `BUY_INSURANCE` column, encode no as 0, and yes as 1.

Moving on, we checked the correlation between each variable using Spearman correlation and removed variables that have multicollinearity with other variables. If the correlation value is greater than 0.7, the 2 variables are then considered to have high collinearity. Below are the variables with high collinearity (Figure 2).

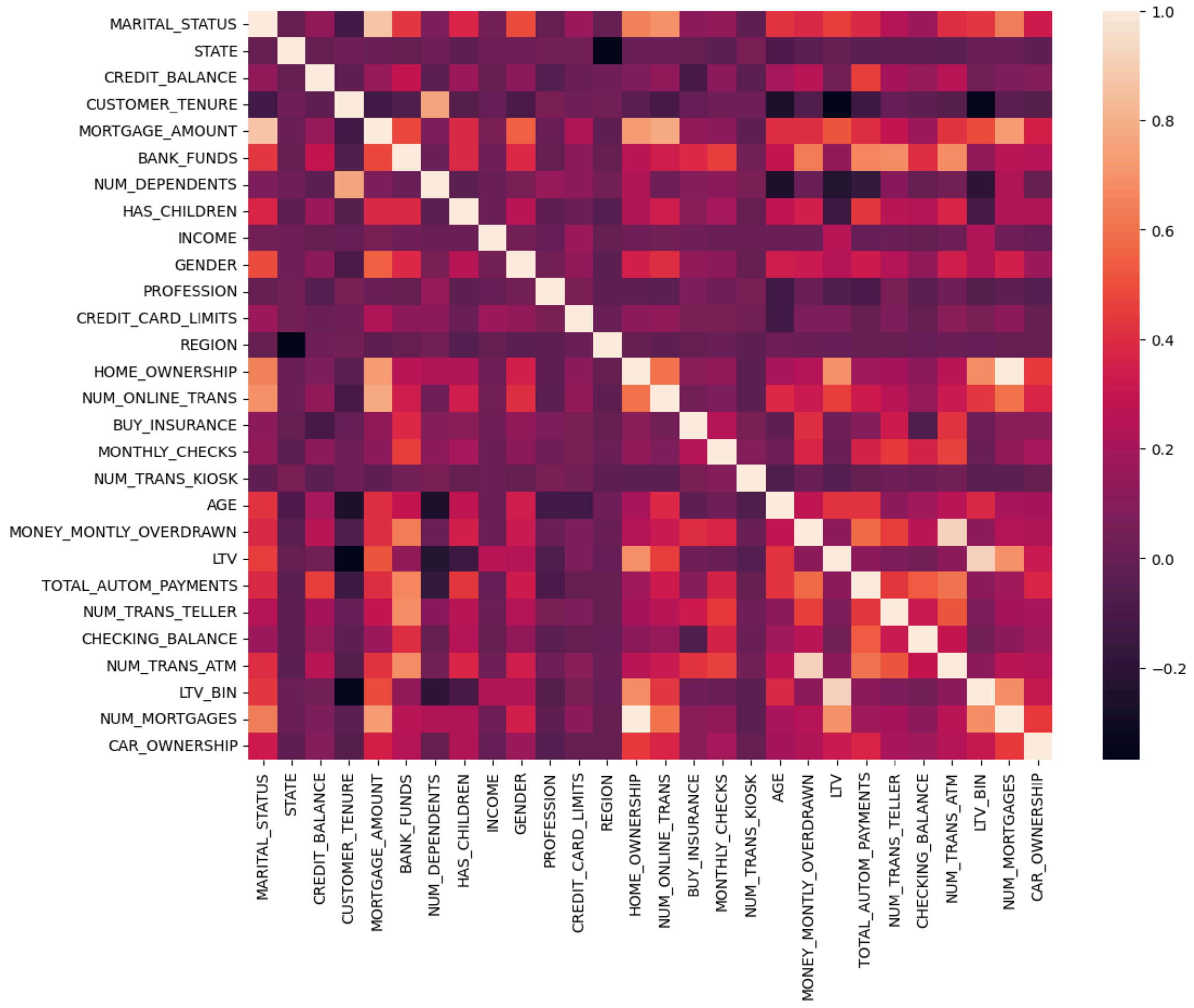


Figure 2: Correlation plot

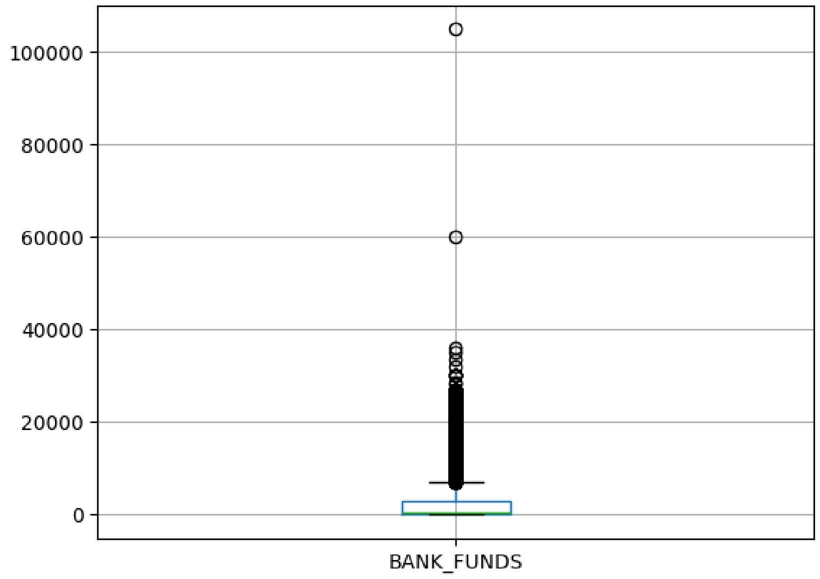
- `MARITAL_STATUS` and `MORTGAGE_AMOUNT` (0.861903)
- `CUSTOMER_TENURE` and `NUM_DEPENDENTS` (0.757758)

- \`MORTGAGE_AMOUNT\` and \`HOME_OWNERSHIP\` (0.730178)
- \`MORTGAGE_AMOUNT\` and \`NUM_ONLINE_TRANS\` (0.770440)
- \`MORTGAGE_AMOUNT\` and \`NUM_MORTGAGES\` (0.730531)
- \`HOME_OWNERSHIP\` and \`NUM_MORTGAGES\` (0.999370)
- \`MONEY_MONTLY_OVERDRAWN\` and \`NUM_TRANS_ATM\` (0.924718)
- \`LTV\` and \`LTV_BIN\` (0.923191)

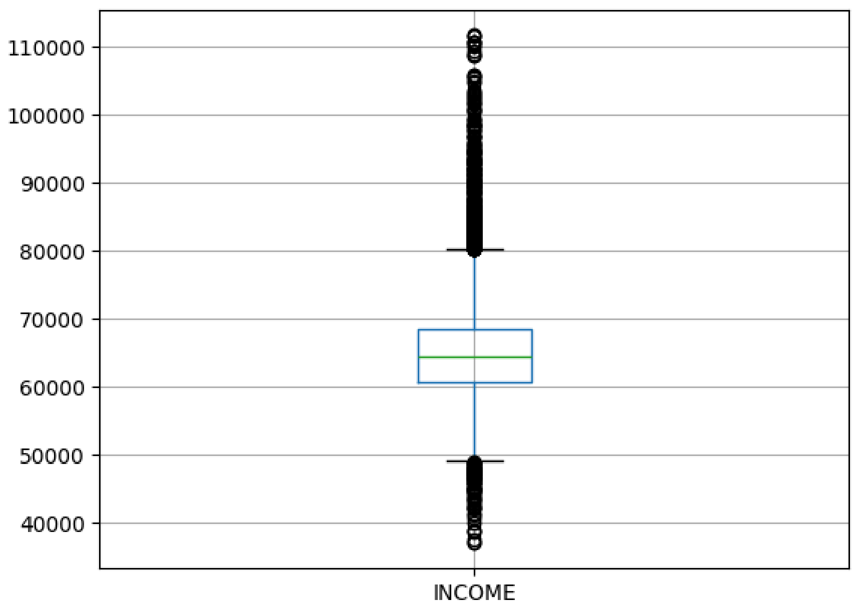
As multicollinearity may inflate the variance and standard error of coefficient estimates [2], we removed some variables that have multicollinearity with other variables, that is \`MORTGAGE_AMOUNT\`, \`NUM_DEPENDENTS\`, \`NUM_MORTGAGES\`, \`MONEY_MONTLY_OVERDRAWN\` and \`LTV_BIN\`.

Next, we did the boxplot analysis to check for outliers. Below are some variables with outliers.

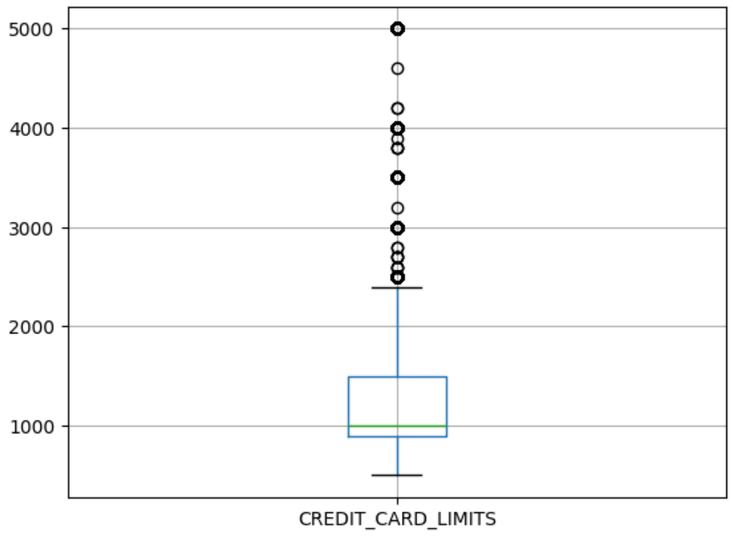
1. \`BANK_FUNDS\`



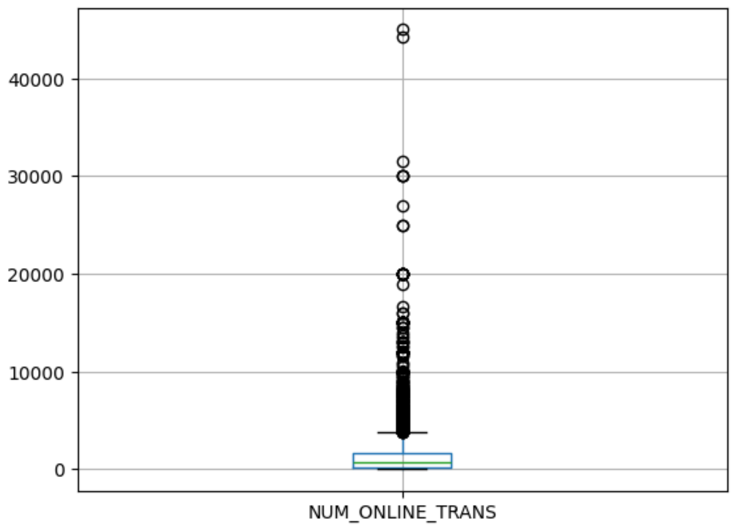
2. \`INCOME\`



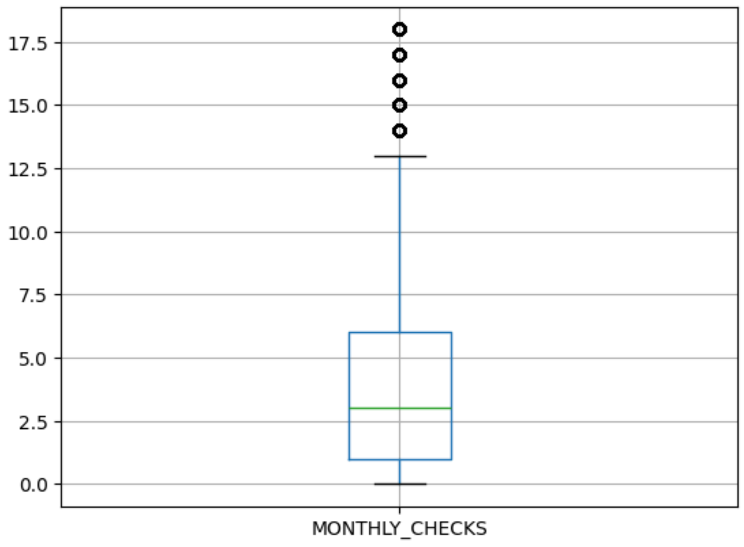
3. `CREDIT_CARD_LIMITS`



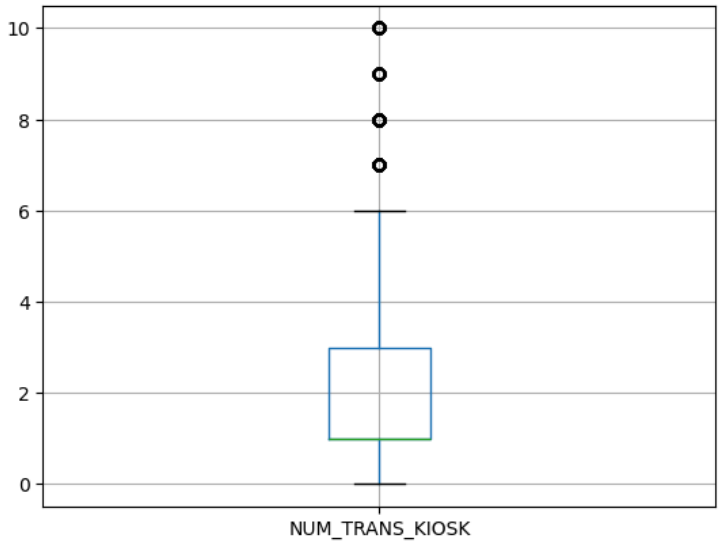
4. `NUM_ONLINE_TRANS`



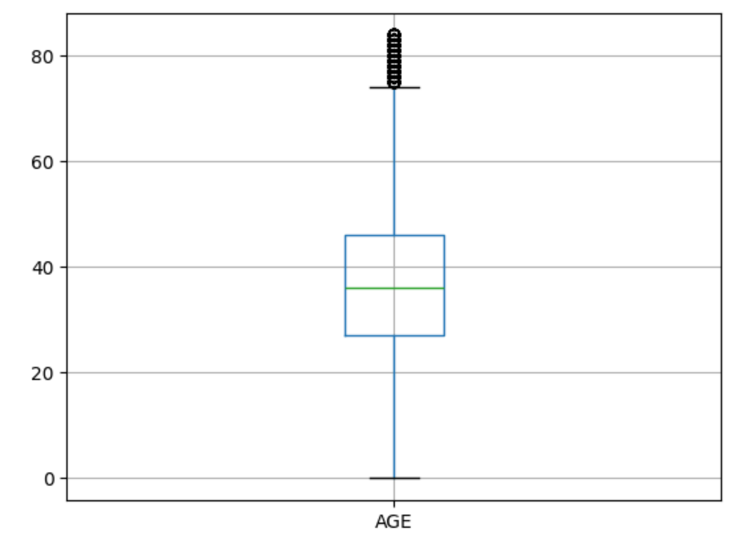
5. \`MONTHLY_CHECKS\`



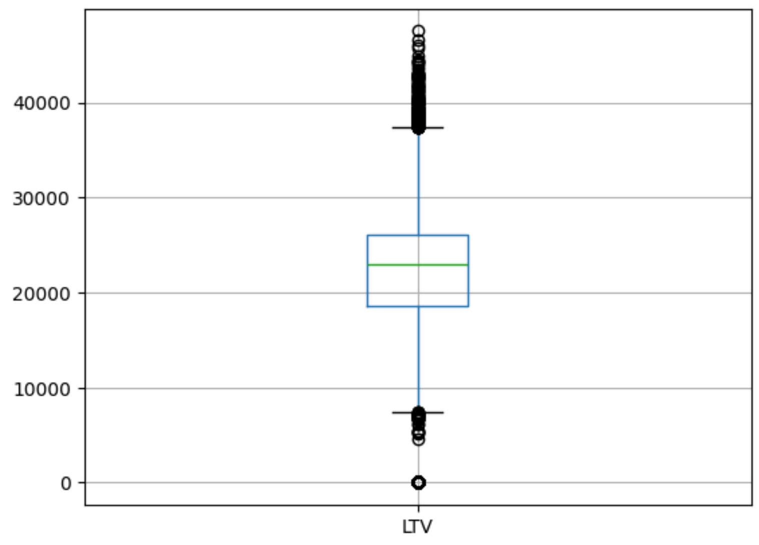
6. \`NUM_TRANS_KIOSK\`



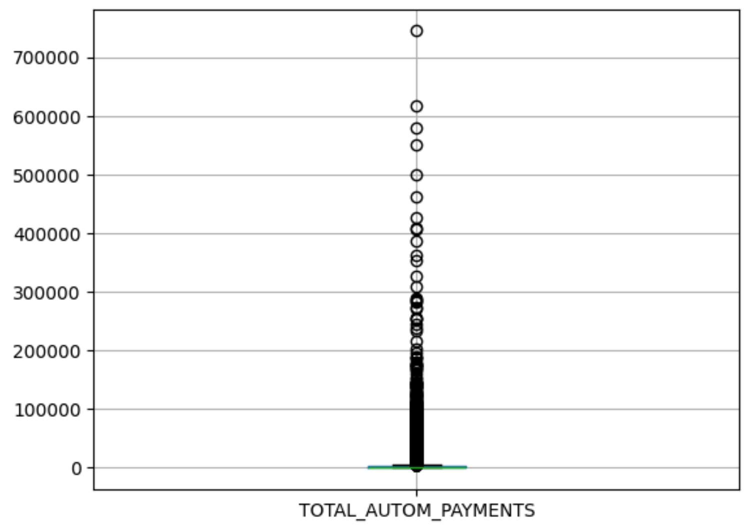
7. \`AGE\`



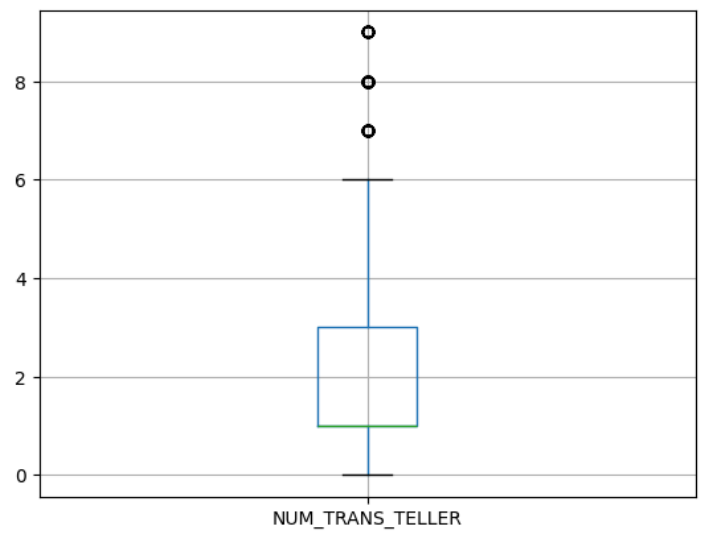
8. `LTV`



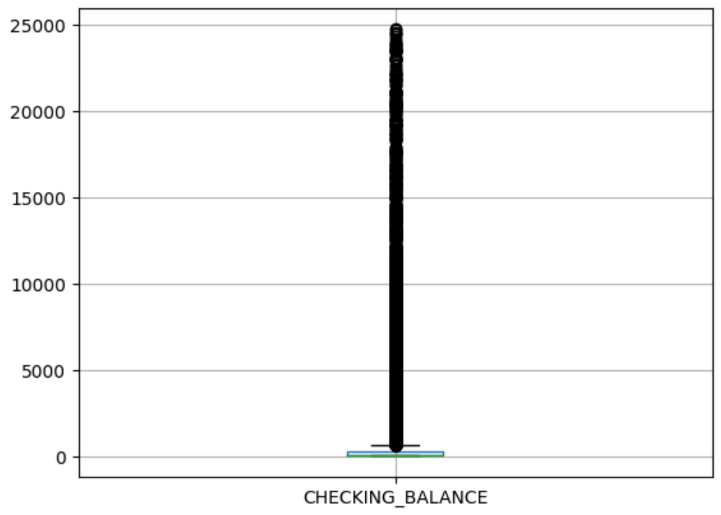
9. `TOTAL_AUTOM_PAYMENTS`



10. `NUM_TRANS_TELLER`



11. `CHECKING_BALANCE`



Removing outliers is a crucial step, as outliers often lead to biased outcomes. Therefore, we took the following steps.

- Remove rows with `MONTHLY_CHECKS` greater than 13.
- Remove rows with `NUM_TRANS_KIOSK` greater than 7.
- Remove rows with `NUM_TRANS_TELLER` greater than 7.
- Remove rows with `CREDIT_CARD_LIMITS` greater than 2400.
- Remove rows with `AGE` greater than 77.
- Remove the `CHECKING_BALANCE` column.
- Remove the `TOTAL_AUTOM_PAYMENTS` column.
- Remove the `NUM_ONLINE_TRANS` column.
- Keep the rest of the columns because those are significant variables, even though they may have a lot of outliers.

Finally, we did feature scaling to our dataset using MinMax Scaling. This is to standardise features in a certain range to handle highly varying magnitudes or values or units. Without feature scaling, bigger-scale features could dominate the learning, producing skewed outcomes [3].

Artificial Intelligence Technologies

Since the main objective of this machine learning project is to predict whether a customer will buy insurance or not, we should perform classification as the output will either be yes or no. There are many classification models, and we are going with logistic regression, perceptron and multi-layer perceptron.

First, we split the dataset into a train set (60%), test set (20%) and validation set (20%). The train set is used to input into the model for training. After training the model, the test set is then used to test the performance of the trained model, that is the accuracy, precision, recall, f1 score and log loss (binary cross-entropy) [3]. The best model will have the highest accuracy, precision, recall and f1 score while having the lowest log loss score.

Then, the validation set is also passed through the model and used to compare its performance against the train set. This ensures that the model is generalized and not overfitted. Ideally, the differences in accuracy and log loss between the validation set and the train set should be minimal. [4]

Solution Evaluation

Below are the performances for each model (Table 1).

Table 1: Model performance

	Logistic Regression	Perceptron	Multi-Layer Perceptron
Accuracy	77.7725%	66.5408%	80.6041%
F1 score	75.9041%	68.8247%	80.4696%
Precision	77.7725%	66.5408%	80.6041%
Recall	75.6514%	78.4138%	80.3532%
Log Loss	8.011590725	12.0599105	6.991005919

Based on the table, we can see that multi-layer perceptron has the highest accuracy, f1 score, precision and recall, followed by logistic regression, while perceptron has the lowest f1 score, precision and recall.

If we compare the loss function, multi-layer perceptron has the lowest log loss, followed by logistic regression, while perceptron has the highest log loss.

Therefore, multi-layer perceptron is the best model for predicting whether a customer will buy insurance or not.

Next, we will check whether this "best model" is actually generalised, or is it overfitted. We will do that by comparing the accuracy and log loss of the train set and the validation set. Below are the graphs of training vs validation loss(Figure 3a) and training vs validation accuracy (Figure 3b).

Figure 3a: Training vs validation loss graph

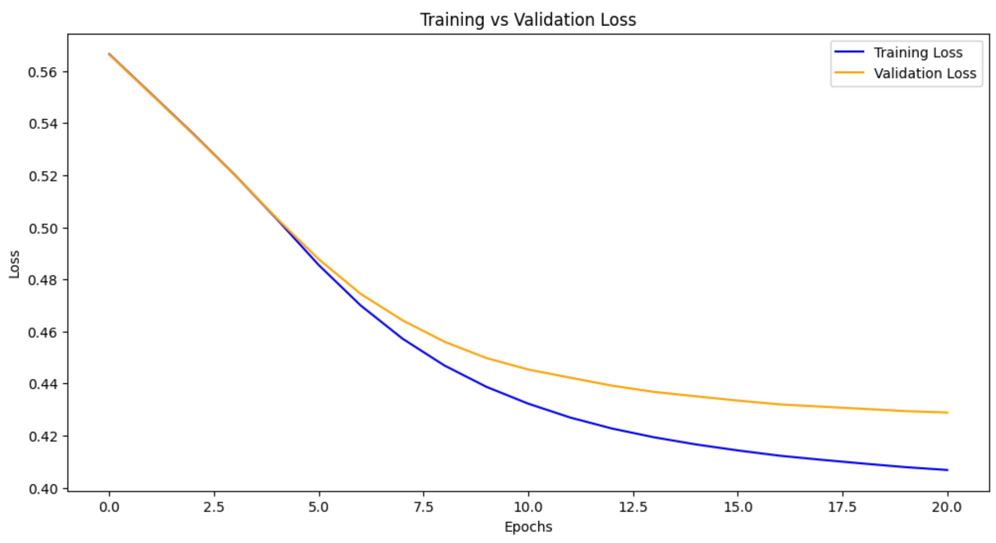
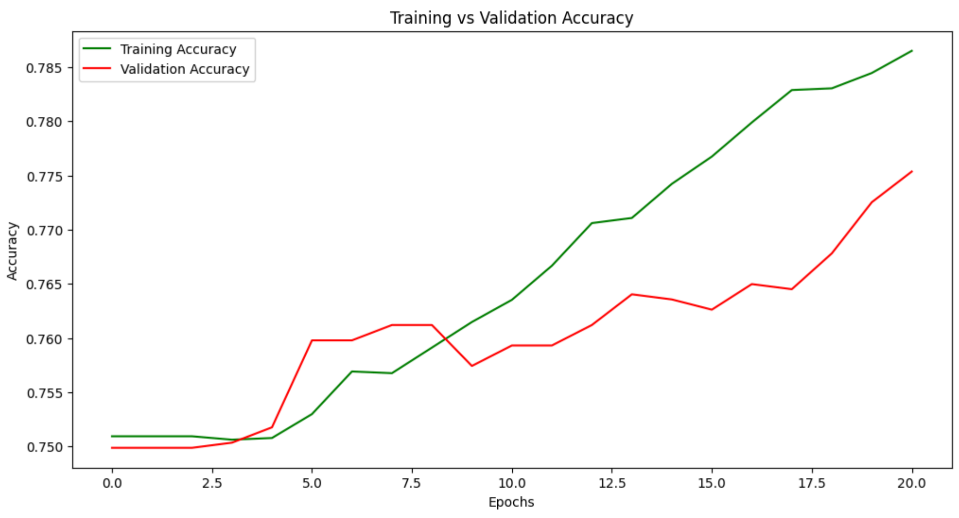


Figure 3b: Training vs validation accuracy graph



We can see that after 20 epochs, the differences of accuracy and log loss between the train set and validation set are not a lot, therefore we can safely assume that the best model is generalised, and is not overfitted, which makes multi-layer perceptron the best model for predicting whether a customer will buy insurance or not.

Conclusion

This project aimed to use AI techniques to predict the ‘BUY_INSURANCE’ variable which is either yes for buying insurance or no for not buying it, after cleaning the data, analysing the data producing the graphs, doing the feature engineering, the Machine Learning, evaluating the Machine Learning, and answering the research question.

We conclude that between, the logistic regression, the perceptron, and the multi-layer perceptron, the multi-layer perceptron demonstrates the best predictive performance which is shown by the metrics such as the accuracy, the F1 score, the recall, the precision, the confusion matrix, the log loss and the model evaluation.
The result suggests that the multi-layer perceptron is particularly effective for this type of classification problem.

Overall this project showcases the application of Machine Learning to a real-world project, it also showcases the power of Machine Learning to be trained on complex datasets and still give good accuracy rates, therefore by developing this kind of AI insurance companies will be able to make more informed decisions and predict customer behaviour which will increase its sales.

References

[1] - Boy Scouts of Americas (2024). State and Territory Abbreviations [online] Available at: https://www.scouting.org/resources/los/states/

[2] - Nathan Rosidi (2024). A Beginner's Guide to Collinearity: What it is and How it affects our regression model [online] Available at: https://www.stratascratch.com/blog/a-beginner-s-guide-to-collinearity-what-it-is-and-how-it-affects-our-regression-model/

[3] - GeeksforGeeks (2023). Feature Engineering: Scaling, Normalization, and Standardization [online] Available at: https://www.geeksforgeeks.org/ml-feature-scaling-part-2/

[4] - Analytics Vidhya (2024). Evaluation Metrics for Classification Model [online] Available at: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/

[5] - GeeksforGeeks (2021). Training vs Testing vs Validation Sets [online] Available at: https://www.geeksforgeeks.org/training-vs-testing-vs-validation-sets/

Appendix

# Artificial Intelligence Group CA
<h2>Members</h2>
* Israa Atike (D00262160)
* Yu Kang Ong (D00262134)
* Kaylee Jacqueline Wijaya (D00262128)

In this project, we will be looking at a dataset called [`CUSTOMER_INSURANCE_LTV`](https://github.com/AnalyticsandDataOracleUserCommunity/MachineLearning/tree/master/cust_insur_ltv), which includes a list of customers' information related to insurance. The main goal of this project is to implement machine learning to predict whether a customer will buy insurance or not based on their demographic, financial, and behavioral information provided in the dataset.

## Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

## Data Loading

```
df = pd.read_csv("CUSTOMER_INSURANCE_LTV.csv")
df
# Checking dimensions
df.shape
# Checking data types
df.dtypes
```

## Data Cleaning
### Removing Columns

Since we are trying to predict whether a customer will buy an insurance or not, the dependent variable should be `BUY_INSURANCE`.
Since `CUSTOMER_ID`, `FIRST_NAME` and `LAST_NAME` are not important variables in classifying `BUY_INSURANCE` , we should remove it.

```
df.drop(columns=["CUSTOMER_ID", "FIRST_NAME", "LAST_NAME"], inplace=True)
df.columns
```

### Missing Values Analysis

```
df.isnull().sum().sort_values(ascending=False)
```

There are no missing values in the dataset.

### Renaming Categorical Variables Value

Next, we will and try to rename some of the values of `PROFESSION` to make it easier for data analysis and machine learning part later on.

We shall categorise the `PROFESSION` based on its respective field.

```
df["PROFESSION"].unique()
#map profession based on field using regex
profession_map = {
        r"Nurse|Medical Doctor|Dentist|Veterinarian|Lab Technician": "Healthcare",
        r"Programmer/Developer|IT Staff|Software Engineer|Technical Writer|DBA": "IT",
        r"Fireman|Law Enforcement Officer": "Public Safety",
        r"Professor|School Teacher": "Education",
        r"Sales Representative|Cashier|Waiter/Waitress|Bank Teller": "Customer Service",
        r"Construction Laborer|Plumber|Mason": "Construction and Trades",
        r"Clerical|Administrative Assistant|Administrator|First-line Manager|Secretary":
"Administrative and Clerical",
        r"Truck Driver": "Transportation",
        r"Childcare Worker|Homemaker": "Childcare and Home",
        r"Author|Publisher": "Creative Media",
        r"Accountant": "Finance",
        r"Not specified|PROF-[\d+]": "Undefined"
}

for original, new in profession_map.items():
        df.loc[df["PROFESSION"].str.match(original), "PROFESSION"] = new
df["PROFESSION"].unique()
df["STATE"].unique()
#map the state name
state_map = {
    "CA": "California",
    "NY": "New York",
    "MI": "Michigan",
    "UT": "Utah",
    "OR": "Oregon",
    "FL": "Florida",
    "IL": "Illinois",
    "MS": "Mississippi",
    "TX": "Texas",
    "WA": "Washington",
    "CO": "Colorado",
    "NV": "Nevada",
    "DC": "District of Columbia",
    "NC": "North Carolina",
    "OK": "Oklahoma",
    "MN": "Minnesota",
    "AL": "Alabama",
    "LA": "Louisiana",
    "NM": "New Mexico",
    "OH": "Ohio",
    "WI": "Wisconsin",
    "MO": "Missouri",
    "AK": "Alaska",
    "AZ": "Arizona"
}


for original, new in state_map.items():
    df.loc[df["STATE"] == original, "STATE"] = new
```
### Imputing Undefinded with CategoricalImputer

Next, we are going to replace "Undefined" value in `PROFESSION` with the most frequent value.

```
df['PROFESSION'].value_counts().plot(kind='bar')

plt.xlabel('Profession')
plt.ylabel('Count')
plt.title('Profession Type Counts')
plt.show()
# replace `Undefined` with the most frequent value, which is 'IT'
# make sure to install the correct version of sklearn_pandas
# pip install sklearn-pandas==1.5.0
from sklearn_pandas import CategoricalImputer
imp_mean = CategoricalImputer("Undefined")
df['PROFESSION'] = imp_mean.fit_transform(df['PROFESSION'])

df['PROFESSION'].value_counts().plot(kind='bar')

plt.xlabel('Profession')
plt.ylabel('Count')
plt.title('Profession Type Counts')
plt.show()
```
## Exploratory Data Analysis

We are interested in the `BUY_INSURANCE` column because that is our target/dependent variable.

```
df['BUY_INSURANCE'].value_counts().sort_index().plot(kind='bar')

plt.xlabel('Buy Insurance (Yes/No)')
plt.ylabel('Count')
plt.title('Distribution of Insurance Purchase Status')
plt.show()
```
Creating a data frame with the data where `BUY_INSURANCE` is 'Yes'

```
buy = df['BUY_INSURANCE'] == 'Yes'
```

```
df_buy = df[buy]
df_buy
### EDA of Social Factors
df_buy['MARITAL_STATUS'].value_counts().plot(kind='bar')
plt.title('Marital Status of People who bought Insurance')
plt.xlabel('Marital Status')
plt.ylabel('Count')
plt.show()
df_buy['GENDER'].value_counts().plot(kind='bar')
plt.title('Gender Distribution of People who Bought Insurance')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
ax = df_buy['HAS_CHILDREN'].value_counts().plot(kind='bar')
plt.title('Distribution of People who Bought Insurance based on Children')
plt.xlabel('Has Children')
plt.ylabel('Count')

ax.set_xticklabels(['Yes', 'No'])
plt.show()
df_buy['NUM_DEPENDENTS'].value_counts().sort_index().plot(kind='bar')
plt.title('Number of Dependents Distribution')
plt.xlabel('Number of Dependents')
plt.ylabel('Count')
plt.show()
plt.figure(figsize=(10, 6))
df_buy['AGE'].value_counts().sort_index().plot(kind='bar')

plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Distribution in Buy Insurance Cases')
plt.show()
df_buy['CAR_OWNERSHIP'].value_counts().sort_index().plot(kind='bar')

plt.xlabel('Cars Owned')
plt.ylabel('Count')
plt.title('Car Ownership Distribution in Buy Insurance Cases')
plt.show()
df_buy['HOME_OWNERSHIP'].value_counts().sort_index().plot(kind='bar')

plt.xlabel('House Owned')
plt.ylabel('Count')
plt.title('Home Ownership Distribution in Buy Insurance Cases')
plt.show()
```

```
### EDA of Demographic factors
df_buy['REGION'].value_counts().sort_values(ascending=True).plot(kind='barh')
plt.title('Distribution of Customers by Region')
plt.xlabel('Count')
plt.ylabel('Region')
plt.show()
df_buy['STATE'].value_counts().sort_values().plot(kind='barh',
edgecolor='black',figsize=(10, 6))
plt.title('Distribution of Customers by State')
plt.xlabel('Count')
plt.ylabel('State')
plt.show()

### EDA of Behavioral Factors
df_buy['CUSTOMER_TENURE'].value_counts().sort_index().plot(kind='bar',
edgecolor='black', figsize=(8, 6))
plt.title('Distribution of Customer Tenure')
plt.xlabel('Customer Tenure (Years)')
plt.ylabel('Count')
plt.show()
df_buy['PROFESSION'].value_counts(ascending=True).plot(kind='barh', edgecolor='black',
figsize=(10, 6))
plt.title('Distribution of Professions')
plt.xlabel('Count')
plt.ylabel('Profession')
plt.show()
plt.hist(df_buy['NUM_ONLINE_TRANS'], bins=20, color='blue', alpha=0.7,
edgecolor='black')

plt.xlabel('Number of Online Transactions')
plt.ylabel('Frequency')
plt.title('Online Transaction Distribution in Buy Insurance Cases')
plt.show()

plt.hist(df_buy['NUM_TRANS_KIOSK'], bins=10, color='blue', alpha=0.7,
edgecolor='black')

plt.xlabel('Number of Kiosk Transactions')
plt.ylabel('Frequency')
plt.title('Kiosk Transaction Distribution in Buy Insurance Cases')
plt.show()
plt.hist(df_buy['NUM_TRANS_TELLER'],bins=9, color='blue', alpha=0.7,
edgecolor='black')
```

```
plt.xlabel('Number of Teller Transactions')
plt.ylabel('Frequency')
plt.title('Teller Transaction Distribution in Buy Insurance Cases')
plt.show()
plt.hist(df_buy['NUM_TRANS_ATM'], bins=8,color='blue', alpha=0.7, edgecolor='black')

plt.xlabel('Number of ATM Transactions')
plt.ylabel('Frequency')
plt.title('ATM Transaction Distribution in Buy Insurance Cases')
plt.show()
df_buy['MONTHLY_CHECKS'].value_counts().sort_index().plot(kind='bar')

plt.xlabel('Number of Monthly Checks')
plt.ylabel('Frequency')
plt.title('Monthly Checks Distribution in Buy Insurance Cases')
plt.show()
### EDA of Financial Factors
df_buy['CREDIT_BALANCE'].plot(kind='hist', bins=30, edgecolor='black', figsize=(8, 6))
plt.title('Distribution of Credit Balance')
plt.xlabel('Credit Balance')
plt.ylabel('Frequency')
plt.show()
df_buy['MORTGAGE_AMOUNT'].plot(kind='hist', bins=30, edgecolor='black', figsize=(8,
6))
plt.title('Distribution of Mortgage Amounts')
plt.xlabel('Mortgage Amount')
plt.ylabel('Frequency')
plt.show()
df_buy['BANK_FUNDS'].plot(kind='hist', bins=30, edgecolor='black', figsize=(8, 6))
plt.title('Distribution of Bank Funds')
plt.xlabel('Bank Funds')
plt.ylabel('Frequency')
plt.show()
df_buy['INCOME'].plot(kind='hist', bins=30,edgecolor='black', figsize=(8, 6))
plt.title('Distribution of Income')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
df_buy['CREDIT_CARD_LIMITS'].plot(kind='hist', bins=20, edgecolor='black', figsize=(8,
6))
plt.title('Distribution of Credit Card Limits')
plt.xlabel('Credit Card Limits')
plt.ylabel('Frequency')
plt.show()
plt.hist(df_buy['MONEY_MONTLY_OVERDRAWN'], bins=20, color='blue', alpha=0.7,
edgecolor='black')

plt.xlabel('Amount of Money Overdrawn Monthly')
plt.ylabel('Frequency')
plt.title('Money Overdrawn Monthly Distribution in Buy Insurance Cases')
plt.show()
plt.hist(df_buy['LTV'], bins=20, color='blue', alpha=0.7, edgecolor='black')

plt.xlabel('Loan-to-value')
plt.ylabel('Frequency')
plt.title('LTV Distribution in Buy Insurance Cases')
plt.show()
plt.hist(df_buy['TOTAL_AUTOM_PAYMENTS'], bins=20, color='blue', alpha=0.7,
edgecolor='black')

plt.xlabel('Total Automated Payments')
plt.ylabel('Frequency')
plt.title('Total Automated Payments Distribution in Buy Insurance Cases')
plt.show()
plt.hist(df_buy['CHECKING_BALANCE'], bins=20, color='blue', alpha=0.7,
edgecolor='black')

plt.xlabel('Balance in Checking Account')
plt.ylabel('Frequency')
plt.title('Balance in Checking Account Distribution in Buy Insurance Cases')
plt.show()
df_buy['LTV_BIN'].value_counts().sort_index().plot(kind='bar')

plt.xlabel('Loan-to-value ratio')
plt.ylabel('Frequency')
plt.title('LTV Distribution in Buy Insurance Cases')
plt.show()
df_buy['NUM_MORTGAGES'].value_counts().sort_index().plot(kind='bar')

plt.xlabel('Number of Mortages')
plt.ylabel('Frequency')
plt.title('Number of Mortages Distribution in Buy Insurance Cases')
plt.show()
## Feature Engineering
### Converting categorical features to numeric
# Encoding Male as 0, Female as 1
df['GENDER'] = df['GENDER'].replace(to_replace = 'M', value=0)
df['GENDER'] = df['GENDER'].replace(to_replace = 'F', value=1)
```

```
#Using one hot encoding for profesion , state  and region
df['STATE'] = df['STATE'].astype('category').cat.codes
df['REGION'] = df['REGION'].astype('category').cat.codes
df['PROFESSION'] = df['PROFESSION'].astype('category').cat.codes
df.head()

# ENCODING LTV_BIN AS 0,1,2,3
df['LTV_BIN'] = df['LTV_BIN'].replace(to_replace='LOW', value=0)
df['LTV_BIN'] = df['LTV_BIN'].replace(to_replace='MEDIUM', value=1)
df['LTV_BIN'] = df['LTV_BIN'].replace(to_replace='HIGH', value=2)
df['LTV_BIN'] = df['LTV_BIN'].replace(to_replace='VERY HIGH', value=3)

# ENCODING  MARTIAL_STATUS AS 0,1
df['MARITAL_STATUS']=df['MARITAL_STATUS'].replace(to_replace='SINGLE',value=0)
df['MARITAL_STATUS']=df['MARITAL_STATUS'].replace(to_replace='MARRIED',value=
1)
df['MARITAL_STATUS']=df['MARITAL_STATUS'].replace(to_replace=['DIVORCED',
'WIDOWED', 'OTHER'],value=2)

# ENCODING BUY_INSURANCE AS 0,1
df['BUY_INSURANCE']= df['BUY_INSURANCE'].replace(to_replace='No',value=0)
df['BUY_INSURANCE']= df['BUY_INSURANCE'].replace(to_replace='Yes',value=1)
df.info()
df.head()
### Removing multicolinearity and outliers
# correlation matrix
corrmat = df.corr(method="spearman")
f, ax = plt.subplots(figsize=(12, 9))
corrmat
sns.heatmap(corrmat, xticklabels=True, yticklabels=True);
corrmat
#checking for high colinearity
filtered_corr = corrmat[corrmat > 0.7]
filtered_corr
```

Belows are the variables with high colinearity (>0.7)

* `MARITAL_STATUS` and `MORTGAGE_AMOUNT` (0.861903)
* `CUSTOMER_TENURE` and `NUM_DEPENDENTS` (0.757758)
* `MORTGAGE_AMOUNT` and `HOME_OWNERSHIP` (0.730178)
* `MORTGAGE_AMOUNT` and `NUM_ONLINE_TRANS` (0.770440)
* `MORTGAGE_AMOUNT` and `NUM_MORTGAGES` (0.730531)
* `HOME_OWNERSHIP` and `NUM_MORTGAGES` (0.999370)
* `MONEY_MONTLY_OVERDRAWN` and `NUM_TRANS_ATM` (0.924718)
* `LTV` and `LTV_BIN` (0.923191)

```
# Dropping colinear variables
df = df.drop('MORTGAGE_AMOUNT', axis=1) #colinear with marital status, home
ownership, num_online_trans, num_mortgages
df = df.drop('NUM_DEPENDENTS', axis=1) #colinear with customer tenure
df = df.drop('NUM_MORTGAGES', axis=1) #colinear with home ownership
df = df.drop('MONEY_MONTLY_OVERDRAWN', axis=1) #colinear with num_trans_ATM
df = df.drop('LTV_BIN', axis=1) #colinear with LTV
### Boxplot analysis
for column in df.columns:
   plt.figure()
   df.boxplot([column])
### Removing outliers
```

From the boxplot analysis, we found out some variables with outliers. We should try to reduce outliers so that the predictive model is not bias.

```
df = df[df.MONTHLY_CHECKS<13]
df = df[df.NUM_TRANS_KIOSK<7]
df = df[df.NUM_TRANS_TELLER<7]
df = df[df.CREDIT_CARD_LIMITS<2400]
df = df[df.AGE<77]
df = df.drop('CHECKING_BALANCE', axis=1)
df = df.drop('TOTAL_AUTOM_PAYMENTS', axis=1)
df = df.drop('NUM_ONLINE_TRANS', axis=1)
for column in df.columns:
   plt.figure()
   df.boxplot([column])
### Feature Scaling
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df= pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
df
## Creating test and train data
y = df['BUY_INSURANCE']
x = df.drop('BUY_INSURANCE', axis=1)

print('Target dimentions: ',y.shape)
print('Independent data dimentions: ',x.shape)

import sklearn.model_selection as model_selection
# split into train 0.6, test 0.2, validation 0.2
# Initial split to separate train+validation and test sets
x_train_val, x_test, y_train_val, y_test = model_selection.train_test_split(
    x, y, train_size=0.8, test_size=0.2, random_state=101
)

# Further split train+validation into train and validation sets
x_train, x_val, y_train, y_val = model_selection.train_test_split(
    x_train_val, y_train_val, train_size=0.75, test_size=0.25, random_state=101  # 0.75 x 0.8 =
0.6, 0.25 x 0.8 = 0.2
)

# Print dimensions
print('x_train dimensions: ', x_train.shape)
print('y_train dimensions: ', y_train.shape)
print('x_val dimensions: ', x_val.shape)
print('y_val dimensions: ', y_val.shape)
print('x_test dimensions: ', x_test.shape)
print('y_test dimensions: ', y_test.shape)
## Machine Learning
### Creating the model
from sklearn.metrics import accuracy_score, log_loss, f1_score, recall_score,
precision_score, classification_report, confusion_matrix
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression

classifiers = [
    LogisticRegression(),
    Perceptron(max_iter=1000, tol=1e-3),
    MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)

]

# Logging for Visual Comparison
log_cols=["Classifier", "Accuracy", "Log Loss"]
log = pd.DataFrame(columns=log_cols)
accuracy = []
f1_sc = []
recall = []
precision = []
loss = []

for clf in classifiers:
    clf.fit(x_train, y_train)
    name = clf.__class__.__name__
    print("="*30)
    print(name)

    print('****Results****')
    train_predictions = clf.predict(x_test)

    acc = accuracy_score(y_test, train_predictions)
    print("Accuracy: {:.4%}".format(acc))
    accuracy.append(acc)

    f1 = f1_score(y_test, train_predictions, average='weighted')
    print('F1 score:{:.4%}'.format(f1))
    f1_sc.append(f1)

    recall_sc = recall_score(y_test, train_predictions, average='weighted')
    print('Recall:{:.4%}'.format(recall_sc))
    recall.append(recall_sc)

    pr_score = precision_score(y_test, train_predictions, average='weighted')
    print('Precision:{:.4%}'.format(pr_score))
    precision.append(pr_score)

    ll = log_loss(y_test, train_predictions)
    print("Log Loss: {}".format(ll))
    loss.append(ll)

    print('Confusion matrix')
    print('-'*30)
    print(confusion_matrix(y_test, train_predictions))
    print('Classification report')
    print('-'*30)
    print(classification_report(y_test, train_predictions))

print("="*30)
### Plot the Training Results
```

Since MLP has the highest accuracy and the highest f1 score, we can conclude that it is the best model. However, we should make sure that is it not underfitted or overfitted. Therefore, we will look at the training vs validation loss plot and training vs validation accuracy plot, to determine if the model is generalised or overfitted.

```
# Special Handling for MLPClassifier to log losses and accuracies
if any(isinstance(clf, MLPClassifier) for clf in classifiers):
    mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000,
random_state=101)
```

```
train_losses = []
val_losses = []
train_accuracies = []
val_accuracies = []

for epoch in range(21):  # Set number of epochs
    mlp.partial_fit(x_train, y_train, classes=np.unique(y_train))

    # Calculate training loss and accuracy
    train_loss = log_loss(y_train, mlp.predict_proba(x_train))
    train_acc = accuracy_score(y_train, mlp.predict(x_train))

    # Calculate validation loss and accuracy
    val_loss = log_loss(y_val, mlp.predict_proba(x_val))
    val_acc = accuracy_score(y_val, mlp.predict(x_val))

    # Store metrics
    train_losses.append(train_loss)
    val_losses.append(val_loss)
    train_accuracies.append(train_acc)
    val_accuracies.append(val_acc)

# Plot losses
plt.figure(figsize=(12, 6))
plt.plot(train_losses, label="Training Loss", color="blue")
plt.plot(val_losses, label="Validation Loss", color="orange")
plt.title("Training vs Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

# Plot accuracies
plt.figure(figsize=(12, 6))
plt.plot(train_accuracies, label="Training Accuracy", color="green")
plt.plot(val_accuracies, label="Validation Accuracy", color="red")
plt.title("Training vs Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

After comparing the training set and validation set of MLP with 20 epochs, the difference in accuracy and log loss is not a lot, implying that the model is generalised and is not overfitted.

Contributions of each student:
Data selection: Israa, kayle and Yu Kang
Data cleaning: Yu kang
Data analysis and feature engineering: Israa and Kaylee
Machine learning: Israa, kaylee and Yu kang.
Report: Israa, kayle and Yu Kang