Predicting Customers' Life Insurance Product using Machine Learning

Chukwuekwu Musei

1.0 Introduction

1.1 Background

The growth indices experienced in the insurance industry in recent time has considerably shown some level of prospect in the future of the sector. According to McKinsey & Company (2013) , customers who patronise insurance companies can be categorized into loyalists and shoppers. While the former remains with the insurer during price variation in the market, the latter has the tendency to switch to the insurer’s competitors (Mau *et al*., 2018). As a result, insurance companies are analysing the customer metrics they have in their possession to predict which of their present or future customers are likely to purchase a life insurance product (Harrison and Ansell, 2002), using data mining approaches for important decision making strategies for their companies (Hung *et al*., 2019), particularly when only 9% of insurance companies have deployed the use of predictive modelling for effective target marketing (Earnix, 2013).

The purpose of this paper is to use three classification methods - logistic regression (LR), Support Vector Machine (SVC) and Random Forest (RF) to predict new customers that will subscribe to a life insurance product. This will aid in targeting the right customers when deploying marketing campaign at low cost (Lau *et al*., 2004; Moro *et al.* 2011).

The paper has been organised in the following way. Session 2 provides the methodology used in carrying out the task. In session 3, the results and findings are interpreted and discussed, while session 4 concludes by highlighting the benefits and limitations of the models.

1.0 *Related Work*

1.2.1 Bigdata Analytics

The emergence of big data has made it possible for financial industry to make sense of customer data (Chen *et al*. 2014), which enabled Mau *et al*. (2018) to perform predictive analytics on customer data in the insurance industry. In their study, they used Random Forest (RF) classification technique to predict how data in the form of online quotes from the insurer’s website was transformed to predict customers who are likely to subscribe or churn to an insurance product. Miguéis *et al.* (2017) and Asare-Frempong and Jayabalan (2017) show the prediction of customer response to bank subscription using random forest (RF), which did better compared to LR, NN and SVM. Their work focused on the imbalance of the data class using synthetic minority oversampling technique and easy ensemble to balance the distribution (Wankhede *et al*., 2019).

This paper will use the LR, RF and SVC for prediction of the Imperials Limited dataset because, 1) LR has the ability to fit models that humans can grasp for easy interpretation (Moro *et al*., 2014), in particular where the LR accuracy is close to other complex classification models (Kuhn and Johnson, 2013), 2) RF will be compared with LR because it is the best method that is more appropriate to predict the behavioral attitude of consumers (Lemmens and Croux 2006), 3) SVM, the third classification model to be used was considered based on its performance against Naïve Bayes (NB), Decision Tree(DT) when (Wankhede *et al*. (2019) performed a study on a Portuguese bank dataset whether customers will subscribe to term-deposit, which is similar to the life insurance product that this paper will address.

2.0 Methodology

The Cross Industry Standard Procedure for Data Mining (CRISP-DM) framework in Figure 1 was used to carry out this task. This method was implemented because it is an industry standard that follows an iterative process of a project's life cycle from understanding a business problem to deployment (Schröer *et al.*,2021).

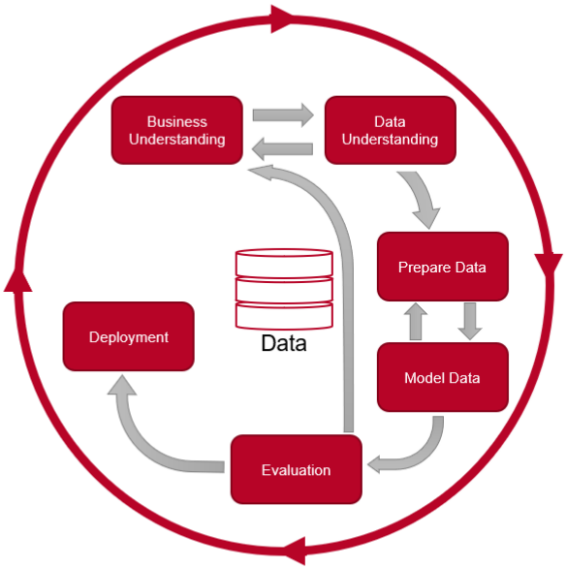


Figure 1: The CRISP-DM stages of a projects lifecycle Source: Schröer *et al.*, 2021

The exploratory data analysis (EDA), preprocessing and cleaning of the Imperial limited dataset was carried out in python. During summary statistics of the insurer's dataset in python, there was a total of 40,000 observations and 13 variables. The bar chart and boxplot in matplotlib.pyplot were used to visualise the dataset to check data quality issues, such as outliers, missing data and some categorical data errors. As a result, the variables with data quality issues were either fixed by coding them as NAs and replacing them by the modal value(see Appendix A). These were done because some of the variables had missing values below 50% of the total observation and if dropped entirely may likely affect the result of the model during classification, particularly when these variables are significant to the target variable for the hypotheses testing.

Additionally, a correlation matrix was used to get insight into the relationships between the predictors, particularly the variables in the hypotheses. These were observed as a prelude to see how the independent variables may likely influence the target variable during modelling (see Appendices A and B). For the prediction of the model, three classification methods were used: LR, RF and SVM. The choice of these models were based on extant literature by (Wankhede *et al.*, 2019) as earlier mentioned above. The cleaned dataset was partitioned into train and test data at 80% and 20% respectively. The test data was used to check for the accuracy of the predicted train dataset in order to evaluate its performance by safeguarding it from overfitting to produce optimally good models (Graham *et al.*, 2018).

3.0 Results and Discussions

3.1 Descriptive Statistics

Tables 1 show the summary statistics of the independent variables in the sales dataset of Imperial Limited. It can be seen in the table and the boxplot in Figure 2 that the average age group is group 4, that is those in the age range of <45 years old. Before now, the age variable underwent data cleaning which rectified the categorical errors in the naming convention that would not allow for a descriptive statistics to be performed, including the machine learning classification modelling. The sales dataset comprised of 40,000 observations and 13 variables, of which there were 7696 missing values in house_val, 2740NAs in marriage variable, 3377 missing observation in the house_owner column, including education variable with 471 missing values and finally, the child variable with 127 NAs, bringing it to a total of 14401 missing observations, equivalent of36% of the total observation. These variables were replaced with the mode or median values depending on the situation of that variable. It can be argued that not outrightly dropping these missing observations improved the accuracy of the classification models used for prediction.

TABLE 1: Result showing descriptive statistics of the sales dataset.

	gender	**education**	**house_val**	**age**	**online**	**marriage**	**child**	**occupation**	**mortgage**	**house_owner**	**region**	**fam_income**
count	40000	40000	40000	40000	40000	40000	40000	40000	40000	40000	40000	40000
mean	0.4783	2.114375	0.030721	4.1138	0.68298	0.87295	0.88305	3.06645	1.38753	0.815225	1.1915	8.173825
std	0.554154	1.189844	0.042221	1.93274	0.46532	0.33303	0.73215	1.115018	0.7105	0.38812	1.1566	2.733212
min	0	0	0	1	0	0	0	0	1	0	0	1
25%	0	1	0.008066	3	0	1	0	2	1	1	0	7
50%	0	2	0.021487	4	1	1	1	3	1	1	1	9
75%	1	3	0.039376	6	1	1	1	4	2	1	2	10
max	2	4	1	7	1	1	2	5	3	1	4	13

On the other hand, occupation, mortgage, region, fam_income, gender and age were encoded numeric values in other to allow for the success of the prediction, which does not recognise categorical variables in the model. In addition, the mortgage, education and fam_income variables were mapped in order to retain its ordinal characteristics when it passes through the prediction model. These cleaning of data quality issues produced the descriptive statistics of the sales dataset below. Furthermore, the fam_income standard deviation of 2.733 indicate that the values of the observations within the fam_income variable is highly dispersed.

3.2 Final Visualisations

The boxplot in Figure 2 show the number of people within the age variable who are likely to opt for a life insurance product are less than those that would not. This insight will help Imperial Limited to target the age group who will patronize a life insurance and also devise a strategy that may attract those who would not patronize them.

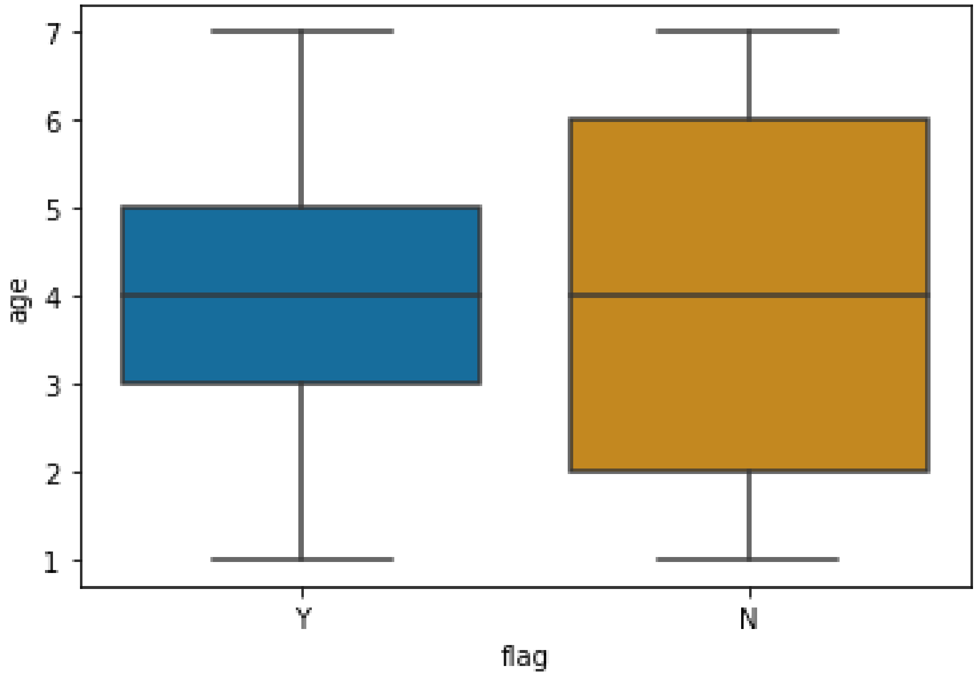


Figure 2: Boxplot showing bivariate relationship with age.

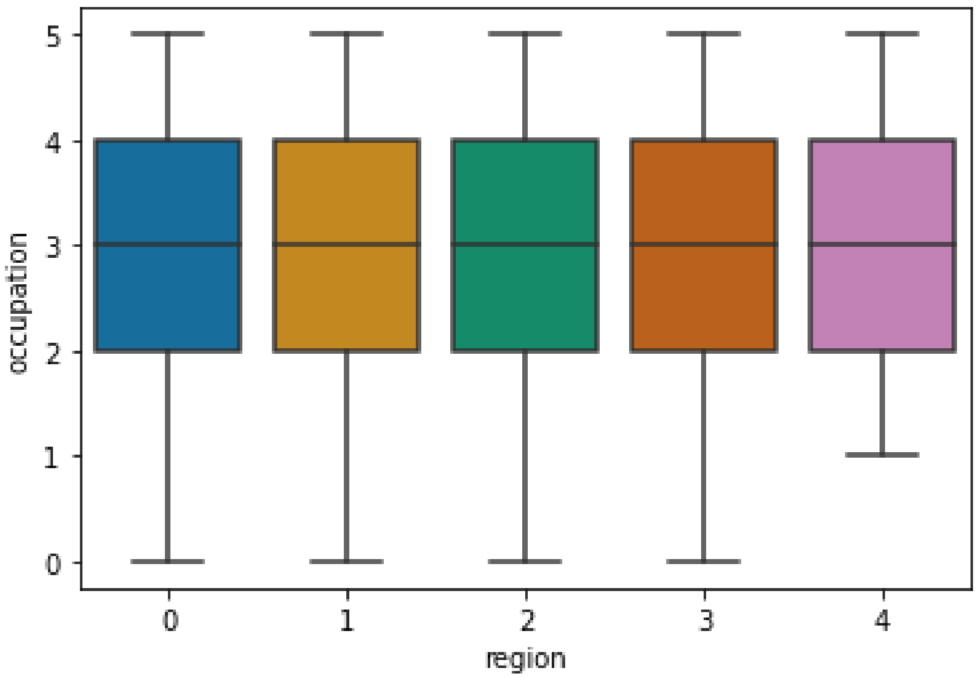


Figure 3: Boxplot showing bivariate relationship between age and occupation.

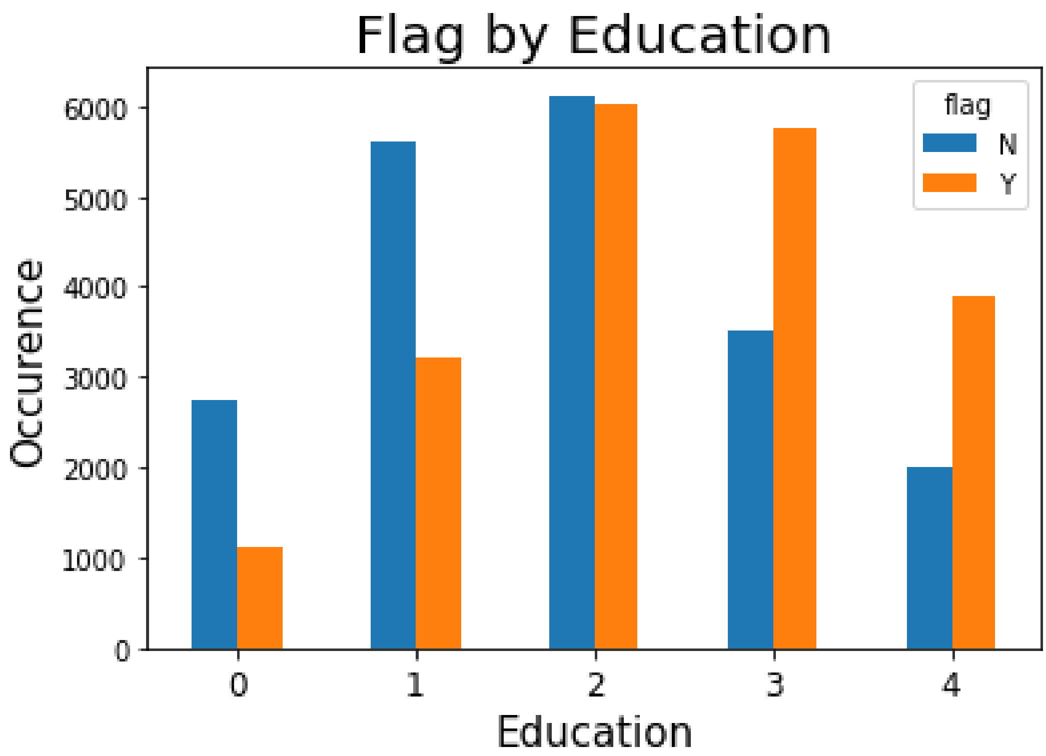


Figure 4: Bar chart showing the bivariate relationship between flag and education.

The bar chart visualization of education variable in Figure 4 show that there is an equal amount of those that will opt for an insurance product and those that will not. This group of people have a bachelor's degree as their educational qualification.

	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_income
gender	1.00	-0.10	-0.08	-0.06	-0.08	-0.02	0.01	0.01	-0.10	-0.08	-0.00	0.11
education	-0.10	1.00	0.24	0.07	0.21	0.05	-0.00	0.28	0.22	0.12	0.02	-0.31
house_val	-0.08	0.24	1.00	0.07	0.13	0.12	-0.02	0.16	0.33	0.17	0.01	-0.39
age	-0.06	0.07	0.07	1.00	0.16	0.17	-0.25	-0.23	0.01	0.17	0.01	-0.09
online	-0.08	0.21	0.13	0.16	1.00	0.17	-0.12	0.11	0.17	0.21	0.03	-0.24
marriage	-0.02	0.05	0.12	0.17	0.17	1.00	-0.10	-0.01	0.13	0.31	-0.04	-0.18
child	0.01	-0.00	-0.02	-0.25	-0.12	-0.10	1.00	0.09	0.05	-0.10	-0.01	-0.01
occupation	0.01	0.28	0.16	-0.23	0.11	-0.01	0.09	1.00	0.19	0.05	-0.01	-0.24
mortgage	-0.10	0.22	0.33	0.01	0.17	0.13	0.05	0.19	1.00	0.26	-0.03	-0.35
house_owner	-0.08	0.12	0.17	0.17	0.21	0.31	-0.10	0.05	0.26	1.00	-0.02	-0.25
region	-0.00	0.02	0.01	0.01	0.03	-0.04	-0.01	-0.01	-0.03	-0.02	1.00	-0.03
fam_income	0.11	-0.31	-0.39	-0.09	-0.24	-0.18	-0.01	-0.24	-0.35	-0.25	-0.03	1.00

Figure 5: Correlation matrix showing bivariate relationships between predictors

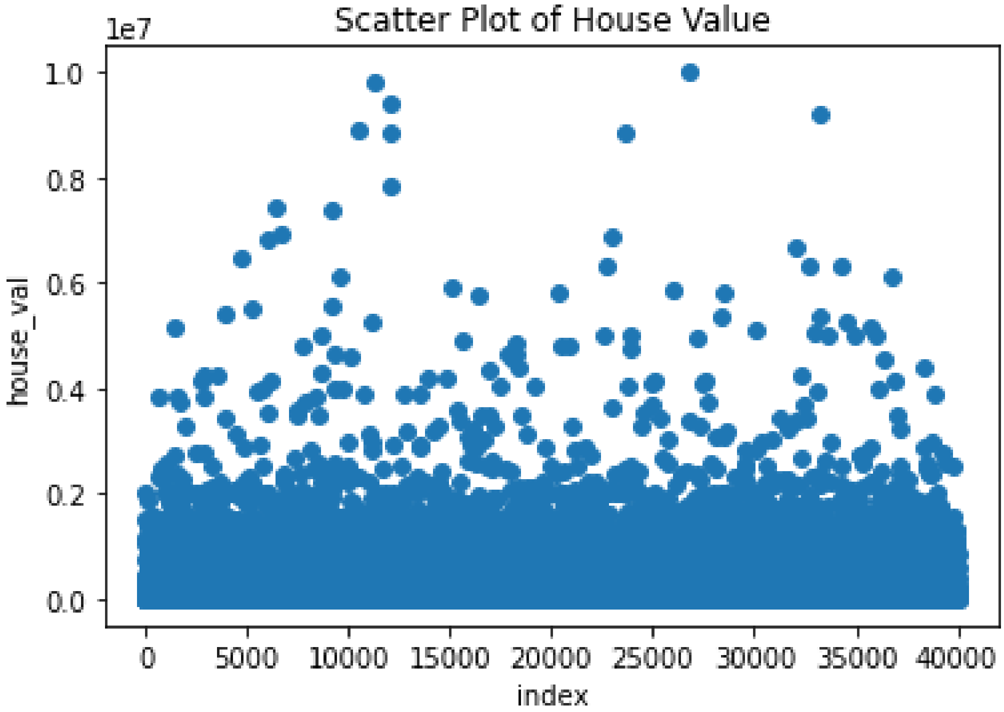


Figure 5: Scatter plot showing the spread for house_val variable.

3.3 Result table for the classification models

	Logistics Regression(LR)	Random Forest(RF)	Support Vector Classifier(SVC)
AUC	0.675	0.675	0.679
Accuracy	67.63	99.49	68.03

Table 3.3 compares the results of three classification models used for predicting the accuracy whether a new customer will buy the life insurance product of Imperials Limited(see Appendix A for details). It can be seen that RF has a high prediction accuracy of 99. 49%, performing better against SVC and LR at 68.03% and 67.63% respectively. The model prediction accuracy and other performance indices in this task gave the same results as the study carried out by Wankhede *et al*. (2019). They performed prediction on a Portuguese bank whether customers are likely to subscribe to term- deposit using LR, RF and SVM. In addition, Kuhn and Johnson (2013) opine that rather than opt for complex models that are difficult to interpret, it is better to use the simple LR model because its accuracy is close to some complex classification model. Taking a look on the above task completed, it can be seen that LR performed well when put side by side with RF in the AUC and sensitivity values. In practice, it can be suggested that LR be a choice for this modelling because it is cost effective and its accuracy is close to other complex models.

3.4 *Confusion Matrix Interpretation*

This table 3.4 is quite revealing in several ways. First, SVC performed better in discriminating between the positive and negative values with an AUC of 0.679, slightly higher than that of LR and RF respectively. Another interesting thing about the result in this table is that the accuracy and AUC in Table 3.3 in all the models are very close to the sensitivity values in Table 3.4 except for RF which has a significant percentage than others in accuracy. This could be as a result of the class balance in all the models, unlike when there is a class imbalance, the result of a high accuracy maybe be misleading as the algorithm will favour the class with more observations.

Table 3.4 Result table for confusion matrix

	Logistics Regression(LR)	**Random Forest(RF)**	**Support Vector Classifier(SVC)**
Sensitivity	0.68	0.68	0.67
Specificity	0.72	0.72	0.50
True Positive	2699	2699	2665
False Negative	1269	1269	1303
False Positive	1328	1328	1260
True Negative	2704	2704	2772

3.4.1 Logistics Regression

$$Sensitivity = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative(FN)}$$

$$= \frac{2699}{2699 + 2704)} = 0.68$$

$$Specificity = \frac{True\ Negative\ (TN)}{True\ Negative\ (TN) + False\ Positive(FP)}$$

$$= \frac{2704}{2704\ + 1328)} = 0.72$$

3.5.2 Random Forest

$$Sensitivity = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative(FN)}$$

$$= \frac{2699}{2699 + 2704)} = 0.68$$

$$Specificity = \frac{True\ Negative\ (TN)}{True\ Negative\ (TN) + False\ Positive(FP)}$$

$$= \frac{2704}{2704 + 1328)} = 0.72$$

3.5.3 Support Vector Classifier

$$Sensitivity = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative(FN)}$$

$$= \frac{2665}{2665 + 1303)} = 0.67$$

$$Specificity = \frac{True\ Negative\ (TN)}{True\ Negative\ (TN) + False\ Positive(FP)}$$

$$= \frac{2772}{2772 + 1260)} = 0.5$$

4.0 Conclusion

This study was designed to see how new customers will respond to a life insurance product. The machine learning classification models of LR, RF and SVC were able to predict accurately on the number of new customers to go for a life insurance product. The relevance of machine learning predicting accurately the performance of the model is clearly supported by the current findings. In practice, it will help Imperial Limited to streamline cost expenses by going for simple models like LR since its level of accuracy is close to some of the complex algorithms. On the other hand, it is crucial to know that one of the significant findings to emerge from this study is that those who have a college and bachelor’s education are beginning to show more prospects to go for a life insurance product. It can be argued that due to the cost of living crises, people are beginning to plan ahead of time in making sure that they have a better life in the future. It could also mean that, this set of age group maybe just leaving the university or just secured a job, and have decided to buy cars or mortgage homes which needs to be insured. Therefore Imperial Limited should design a marketing plan that ensures that they target this age group in their quest to remain profitable as a business as well as continue to be a going concern.

References

Chen, K., Hu, Y. H. & Hsieh, Y. C. (2015) 'Predicting customer churn from valuable B2B customers in the logistics industry: a case study', *Inf Syst E-Bus Manage*, 13, pp. 475–494.Available at: https://doi.org/10.1007/s10257-014-0264-1.

Desai, R., & Khairnar, V. (2022) 'Hybrid prediction model for the success of bank telemarketing. in: Raj, J.S., Palanisamy, R., Perikos, I., Shi, Y. (eds) Intelligent Sustainable Systems. Lecture Notes in Networks and Systems, vol 213. Springer, Singapore. https://doi.org/10.1007/978-981-16-2422-3_54.

Graham, B., Bond, R., Quinn, M., & Mulvenna, M. (2018) 'Using data mining to predict hospital admissions from the emergency departmen*t. IEEE Access, 6, 10458–10469.* Available at: https://doi/10.1109/access.2018.2808843. (Accessed 3 March 2023)

Earnix (2013) 'Insurance Predictive Modeling Survey'. Available at: http://earnix.com/2013-insurance-predictive-modeling-survey/3594/.

Gupta, A., Raghav, A. and Srivastava, S. (2021) 'Comparative Study of Machine Learning Algorithms for Portuguese Bank Data', *International Conference on Computing, Communication, and Intelligent Systems*, pp. 401-406, Available at: https://doi/10.1109/ICCCIS51004.2021.9397083.

Guyon, I. & Elisseeff, A. (2003) 'An introduction to variable and feature selection' *Journal of Machine Learning Research*, 3, pp. 1157-1182.

Hung, P. D., Hanh, T. D. & Tung, T. D. (2019) 'Term deposit subscription prediction using Spark MLlib and ML packages*', Proceedings of the 5th International Conference on E-Business and Applications.Available at:* https://doi/10.1145/3317614.3317618 (Accessed: 5 March 2023).

Harrison, T., & Ansell, J. (2002) 'Customer retention in the insurance industry: using survival analysis to predict cross-selling opportunities*', Journal of Financial Services Marketing,* 6(3), pp 229–239. doi:10.1057/palgrave.fsm.4770054

Kuhn, M. and Johnson, K (2013) *Applied Predictive Modelling*. London, U.K.: Springer.

Lau, Kn., Chow, H. & Liu, C. (2004) 'A database approach to cross selling in the banking industry: practices, strategies and challenges', *J Database Mark Cust Strategy Manag*, 11, pp. 216–234. Available at: https://doi.org/10.1057/palgrave.dbm.3240222.

Lemmens, A. and Croux, C. (2006) 'Bagging and boosting classification trees to predict churn', *Journal of Marketing Research,* Vol. 43(2), pp. 276-286.

Mau, S., Pletikosa, I., & Wagner, J. (2018) 'Forecasting the next likely purchase events of insurance customers*', International Journal of Bank Marketing.* Vol. 36(6), pp. 1125-1144. Available at: https://doi/10.1108/ijbm-11-2016-0180.

McKinsey & Company (2013) ‘Beyond price: the rise of customer-centric marketing in insurance’, availableat:https://www.mckinsey.com/~/media/mckinsey/dotcom/client_service/financial%20services/latest%20thinking/insurance/beyond_price_the_rise_of_customer-centric_marketing_in_insurance.ashx

Miguéis, V.L., Camanho, A.S. & Borges, J. (2017) ‘Predicting direct marketing response in banking: comparison of class imbalance methods’, *Serv Bus*, 11, pp.831–849 https://doi.org/10.1007/s11628-016-0332-3.

Moro, S., Laureano, R. M.S. & Cortez, P. (2011) ‘Using data mining for bank direct marketing: an application of the crisp-dm methodology’ [Online]. Available at: MoroCortezLaureano_DMApproach4DirectMKT.pdf.

Moro, S., Cortez, P., & Rita, P. (2014) ‘A data-driven approach to predict the success of bank telemarketing’, *Decision Support Systems, 62, 22–31.*Available at: https://doi/10.1016/j.dss.2014.03.001.

Schröer, C., Kruse, F., & Gómez, J. M. (2021) ‘A systematic literature review on applying CRISP-DM process model*’, Procedia Computer Science,* 181, pp. 526–534. https://doi/10.1016/j.procs.2021.01.199.

Wankhede, P., Singh, R., Rathod, R., Patil, J. & Khadtare T.D. (2019) ‘Improving prediction of potential clients for bank term deposits using machine learning approaches’, [Online]. Available at: IRJET-V6I5100520190814-54045-zhs1m6-libre.pdf (d1wqtxts1xzle7.cloudfront.net).

Appendix A: R Code

Load in libraries

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

Load in sales dataset

```
#Loading in Sales_data 3

musei = pd.read_csv('/content/drive/MyDrive/Documents/Data Mining/DMI- Assignment 1/sales_data 3.csv')
```

Exploring the sales dataset

```
musei.info() #exploring the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   flag         40000 non-null  object
 1   gender       40000 non-null  object
 2   education    39259 non-null  object
 3   house_val    40000 non-null  int64 
 4   age          40000 non-null  object
 5   online       40000 non-null  object
 6   marriage     25973 non-null  object
 7   child        40000 non-null  object
 8   occupation   40000 non-null  object
 9   mortgage     40000 non-null  object
 10  house_owner  36623 non-null  object
 11  region       40000 non-null  object
 12  fam_income   40000 non-null  object
dtypes: int64(1), object(12)
memory usage: 4.0+ MB
```

```
musei.head(10) #exploring the first 10 observation of the dataset
```

	flag	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_income
0	Y	M	4. Grad	756460	1_Unk	N	NaN	U	Professional	1Low	NaN	Midwest	L
1	N	F	3. Bach	213171	7_>65	N	NaN	U	Professional	1Low	Owner	Northeast	G
2	N	M	2. Some College	111147	2_<=25	Y	NaN	Y	Professional	1Low	Owner	Midwest	J
3	Y	M	2. Some College	354151	2_<=25	Y	Single	U	Sales/Service	1Low	NaN	West	L
4	Y	F	2. Some College	117087	1_Unk	Y	Married	Y	Sales/Service	1Low	NaN	South	H
5	Y	F	3. Bach	248694	6_<=65	Y	Married	N	Professional	2Med	Owner	West	G
6	Y	M	3. Bach	2000000	1_Unk	Y	Married	U	Professional	1Low	NaN	Northeast	C
7	N	F	3. Bach	416925	5_<=55	Y	Married	Y	Professional	1Low	Owner	South	I
8	N	F	1. HS	207676	4_<=45	Y	NaN	Y	Blue Collar	1Low	Renter	West	D

```
educ_nan_count = musei['education'].isna().sum()#Number of missing values in education column
marr_nan_count = musei['marriage'].isna().sum()#Number of missing values in marriage column
ho_nan_count = musei["house_owner"].isna().sum()#Number of missing values in house_owner column


# Printing the number of missing values in education column
# Printing the number of missing values in marriage column
# Printing the number of missing values in house_owner column

print("The number of values missing from the education column is: " + str(educ_nan_count))
print("The number of values missing from the marriage column is: " + str(marr_nan_count))
print("The number of values missing from the house_owner column is: " + str(ho_nan_count))
```

```
The number of values missing from the education column is: 741
The number of values missing from the marriage column is: 14027
The number of values missing from the house_owner column is: 3377
```

Checking each variable to explore and check for data quality issues.

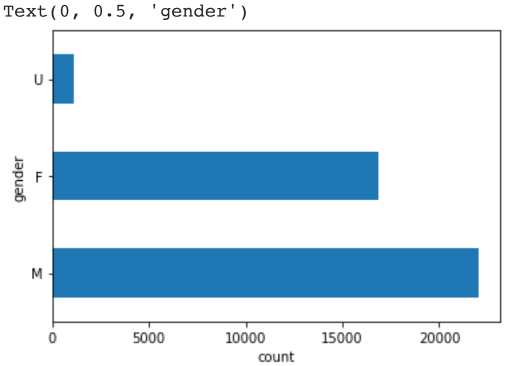
1. First, the gender variable is analysed

```
gend= musei['gender'].value_counts() #checking summary of gender variable
print(gend)
```

```
M    22019
F    16830
U     1151
Name: gender, dtype: int64
```

```
musei.gender.value_counts().plot.barh() #plotting the gender variable for more detailed exploration
plt.xlabel('count')
plt.ylabel('gender')
```

```
Text(0, 0.5, 'gender')
```



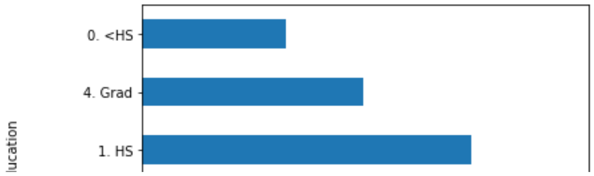
2. Second, the education variable is analysed

```
educ= musei['education'].value_counts() #checking summary of gender variable
print(educ)
```

```
2. Some College    11400
3. Bach             9267
1. HS               8828
4. Grad             5916
0. <HS              3848
Name: education, dtype: int64
```

```
musei.education.value_counts().plot.barh() #plotting the education variable for more detailed exploration
plt.xlabel('count')
plt.ylabel('education')
```

```
Text(0, 0.5, 'education')
```



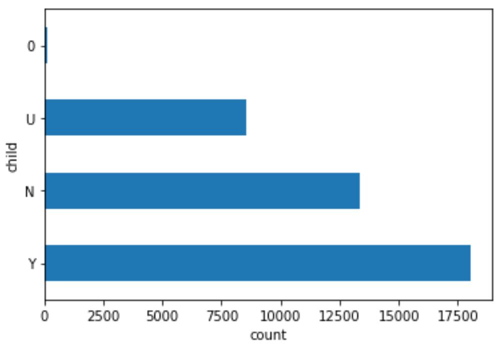
3. Third, the child variable is analysed

```
chd = musei['child'].value_counts() #checking summary of child variable
print(chd)
```

```
Y    18012
N    13333
U     8528
0      127
Name: child, dtype: int64
```

```
musei.child.value_counts().plot.barh() #plotting the child variable for more detailed exploration
plt.xlabel('count')
plt.ylabel('child')
```

```
Text(0, 0.5, 'child')
```



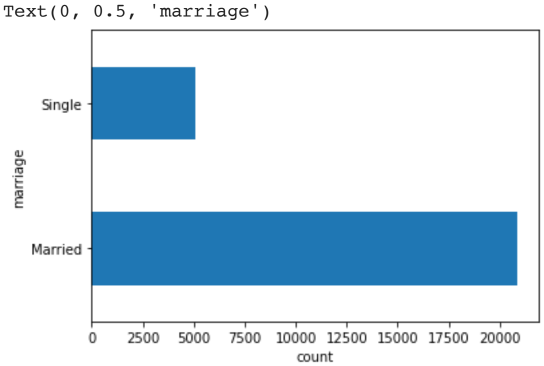
4. Fourth, the marriage variable is analysed

```
marr = musei['marriage'].value_counts() #checking summary of marriage variable
print(marr)
```

```
Married    20891
Single      5082
Name: marriage, dtype: int64
```

```
musei.marriage.value_counts().plot.barh() #plotting the marriage variable for more detailed exploration
plt.xlabel('count')
plt.ylabel('marriage')
```

```
Text(0, 0.5, 'marriage')
```



5. Fifth, the house_value variable is analysed

```
hv = musei['house_val'].value_counts() #checking summary of house_val variable
print(hv)
```

```
0          7696
1000000      96
1500000      51
2000000      36
294300       29
           ...
232394        1
75962         1
297631        1
734006        1
213596        1
Name: house_val, Length: 19572, dtype: int64
```

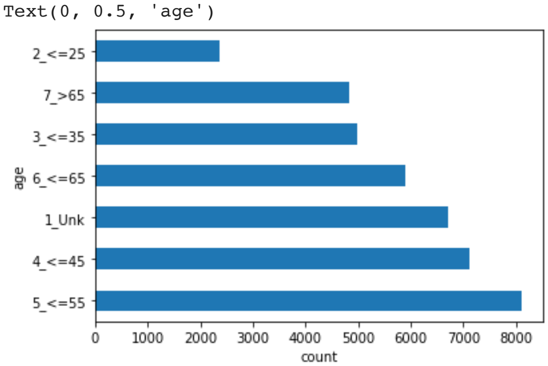
6. Sixth, the Age variable is analysed

```
ag = musei['age'].value_counts() #checking summary of age variable
print(ag)
```

```
5_<=55    8103
4_<=45    7115
1_Unk     6709
6_<=65    5907
3_<=35    4984
7_>65     4822
2_<=25    2360
Name: age, dtype: int64
```

```
musei.age.value_counts().plot.barh() #plotting the age variable for more detailed exploration
plt.xlabel('count')                  #There are 7 different age group in the dataset
plt.ylabel('age')
```

```
Text(0, 0.5, 'age')
```



Data Cleaning.

Gender

```
musei['gender'] = musei['gender'].replace('M', 0)#Replacing 'M' with 0 for Male gender
musei['gender'] = musei['gender'].replace('F', 1)#Replacing 'F' with 1 for Female
musei['gender'] = musei['gender'].replace('U', 2)#Replacing 'U' with 2 for Unknown which could be Transgender or 'prefer not to a
gend_final = musei['gender'].value_counts()
print(gend_final)
```

```
0    22019
1    16830
2     1151
Name: gender, dtype: int64
```

Education

```
#Replacing the 741 missing values in education column with the modal value

mode_educ = musei['education'].mode()[0]
musei['education'] = musei['education'].fillna(mode_educ)
```

```
#mapping of education as ordinal variable

mus_education_mapping = {'0. <HS':0, '1. HS': 1, '2. Some College': 2, '3. Bach': 3, '4. Grad': 4}
musei['education'] =musei['education'].map(mus_education_mapping)
musei.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   flag         40000 non-null  object
 1   gender       40000 non-null  int64 
 2   education    40000 non-null  int64 
 3   house_val    40000 non-null  int64 
 4   age          40000 non-null  object
 5   online       40000 non-null  object
 6   marriage     25973 non-null  object
 7   child        40000 non-null  object
 8   occupation   40000 non-null  object
 9   mortgage     40000 non-null  object
 10  house_owner  36623 non-null  object
 11  region       40000 non-null  object
 12  fam_income   40000 non-null  object
dtypes: int64(3), object(10)
memory usage: 4.0+ MB
```

House_val

```
#Counting the number of zeros in house_val

count = (musei['house_val'] == 0).sum()
print('The number of zeros in house_val column is : ', count)
```

```
The number of zeros in house_val column is :  7696
```

```
musei['house_val'].median() #Finding the median value of the house_val column
```

```
214872.0
```

```
##Replacing the zeros with the median of the house_val column

musei['house_val'].replace(0,musei['house_val'].median())
```

```
0        756460
1        213171
2        111147
3        354151
4        117087
          ...  
39995    214872
39996    213596
39997    134070
39998    402210
39999    836030
Name: house_val, Length: 40000, dtype: int64
```

Age

```
#mapping of age as ordinal variable

mus_age_mapping = {'1_Unk':1, '2_<=25': 2, '3_<=35': 3, '4_<=45': 4, '5_<=55': 5, '6_<=65': 6, '7_>65': 7}
musei['age'] =musei['age'].map(mus_age_mapping)
```

Online

```
musei['online'] = musei['online'].replace("N", 0)                 #encoding the online variable
musei['online'] = musei['online'].replace("Y", 1)
```

Marriage

```
mode_marriage = musei['marriage'].mode()[0]
musei['marriage'] = musei['marriage'].fillna(mode_marriage)       #Replacing the missing values with mode


musei['marriage'] = musei['marriage'].replace("Single", 0)        #Replacing single with 0
musei['marriage'] = musei['marriage'].replace("Married", 1)       #Replacing married with 1
```

Child

```
#Replaced the child variable response and merged 0 to 'unknown'

musei['child'] = musei['child'].replace({'0': 'Unknown', 'N': 'No', 'U': 'Unknown', 'Y': 'Yes'})


musei['child'] = musei['child'].replace("No", 0)                         #encoded 'No' to 0
musei['child'] = musei['child'].replace("Yes", 1)                        #encoded 'Yes' to 1
musei['child'] = musei['child'].replace("Unknown", 2)                    #encoded 'Unknown' to 2
```

Occupation

```
musei['occupation'] = musei['occupation'].replace('Farm', 0)            #Replacing 'Farm' with 0
musei['occupation'] = musei['occupation'].replace('Retired', 1)        #Replacing 'Retired' with 1
musei['occupation'] = musei['occupation'].replace('Blue Collar', 2)    #Replacing 'Blue Collar' with 2
musei['occupation'] = musei['occupation'].replace('Sales/Service', 3)  #Replacing 'Sales/Service' with 3
musei['occupation'] = musei['occupation'].replace('Professional', 4)   #Replacing 'Professional' with 4
musei['occupation'] = musei['occupation'].replace('Others', 5)         #Replacing 'Farm' with 5
```

Mortgage

```
 #mapping of mortgage as numeric variable

mus_mortgage_mapping = {'1Low':1, '2Med':2, '3High':3}
musei['mortgage'] =musei['mortgage'].map(mus_mortgage_mapping)
```

House owner

```
mode_house_owner = musei['marriage'].mode()[0]
musei['house_owner'] = musei['house_owner'].fillna(mode_house_owner)   #Replacing the missing values with mode


musei['house_owner'] = musei['house_owner'].replace("Renter", 0)   #Replacing Renter with 0 i.e. to numeric
musei['house_owner'] = musei['house_owner'].replace("Owner", 1)    #Replacing Owner with 1 i.e. to numeric
```

Region

```
musei['region'] = musei['region'].replace("South", 0)         #Replacing South with 0 i.e. to numeric
musei['region'] = musei['region'].replace("West", 1)          #Replacing West with 1 i.e. to numeric
musei['region'] = musei['region'].replace("Midwest", 2)       #Replacing Midwest with 2 i.e. to numeric
musei['region'] = musei['region'].replace("Northeast", 3)     #Replacing Northeast with 3 i.e. to numeric
musei['region'] = musei['region'].replace("Rest", 4)          #Replacing Rest with 4 i.e. to numeric
```

Family income

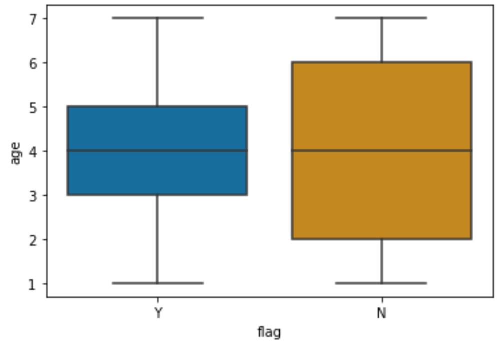
```
#mapping of age variable

mus_family_income_mapping = {'U':1, 'L': 2, 'K': 3, 'J': 4, 'I': 5, 'H': 6, 'G': 7,
                             'F': 8, 'E': 9, 'D': 10, 'C': 11, 'B': 12, 'A': 13}
musei['fam_income'] =musei['fam_income'].map(mus_family_income_mapping)
```

Visualisations

```
#Boxplot showing age group who will opt and not opt for a life insurance product
#with an average mean age 45 years (i.e age group 4)
```

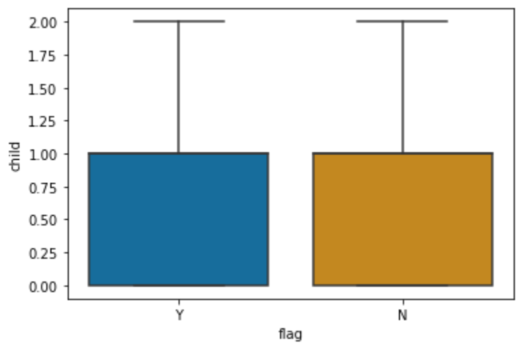
```
sb.boxplot(x='flag',y='age',data=musei, palette='colorblind')
```

<AxesSubplot:xlabel='flag', ylabel='age'>



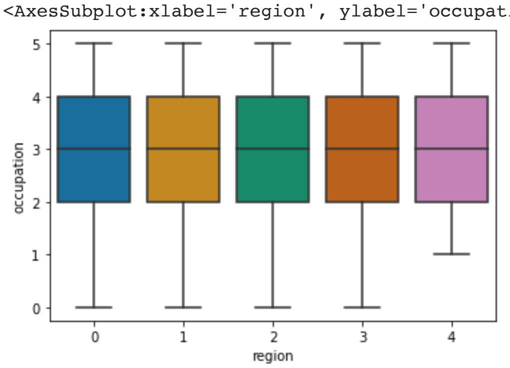

```
#Boxplot showing even distribution of child variable
#who opt and not opt for a life insurance product

sb.boxplot(x='flag',y='child',data=musei, palette='colorblind')
```

<AxesSubplot:xlabel='flag', ylabel='child'>




```
#The bivariate relationship between region and occupation

sb.boxplot(x='region',y='occupation',data=musei, palette='colorblind')
```

<AxesSubplot:xlabel='region', ylabel='occupation'>

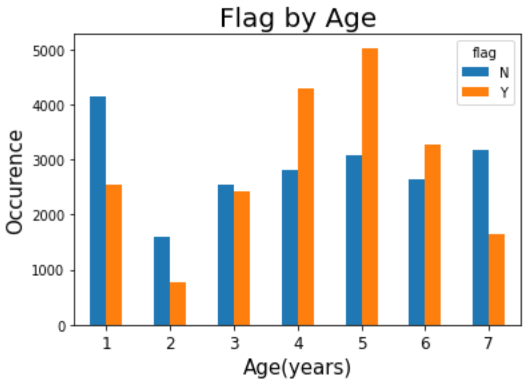



```
#The bivariate result show that male and female gender are evenly distributed

sb.boxplot(x='flag',y='gender',data=musei, palette='colorblind')
```

```
<AxesSubplot:xlabel='flag', ylabel='gender'>
```

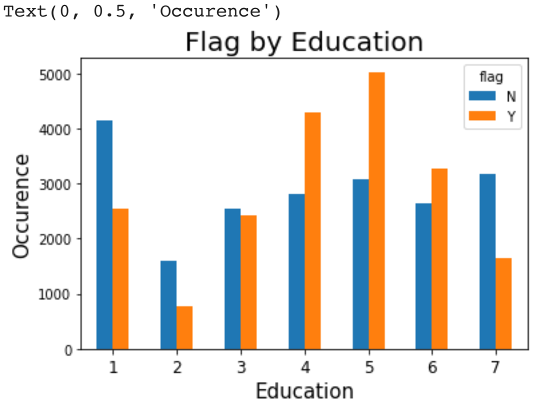

```
#Barchat showing the distributioin of education variable with flag
#Those in a group 5 (<=55) go for the life insurance product
#While age group 2 (<=25) are the least patronage of the life insurance product

pd.crosstab(musei.age,musei.flag).plot(kind='bar')
plt.xticks(rotation=360, fontsize=12)
plt.title('Flag by Age', fontsize=20)
plt.xlabel('Age(years)', fontsize=15)
plt.ylabel('Occurence', fontsize=15)
```

```
Text(0, 0.5, 'Occurence')
```



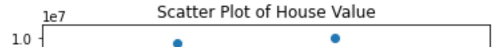

```
#Barchat showing the distributioin of education variable with flag

pd.crosstab(musei.age,musei.flag).plot(kind='bar')
plt.xticks(rotation=360, fontsize=12)
plt.title('Flag by Education', fontsize=20)
plt.xlabel('Education', fontsize=15)
plt.ylabel('Occurence', fontsize=15)
```

```
Text(0, 0.5, 'Occurence')
```




```
#Scatter plot showing the spread of the house_val variable

plt.scatter(musei.index, musei['house_val'])
plt.title('Scatter Plot of House Value')
plt.xlabel('index')
plt.ylabel('house_val')
```

```
Text(0, 0.5, 'house_val')
```



Measures of Association

```
mus_corr = pd.DataFrame(musei)                                          #Correlation matrix of the predictors
corr_matrix = musei.corr()
print(corr_matrix)
```

```
               gender  education  house_val       age    online  marriage  \
gender       1.000000  -0.100791  -0.076044 -0.064150 -0.077823 -0.023338
education   -0.100791   1.000000   0.237863  0.068124  0.206466  0.051941
house_val   -0.076044   0.237863   1.000000  0.067946  0.127037  0.122543
age         -0.064150   0.068124   0.067946  1.000000  0.158649  0.171767
online      -0.077823   0.206466   0.127037  0.158649  1.000000  0.169052
marriage    -0.023338   0.051941   0.122543  0.171767  0.169052  1.000000
child        0.005021  -0.001692  -0.022602 -0.254462 -0.118371 -0.103286
occupation   0.010062   0.282588   0.157311 -0.231376  0.106521 -0.011196
mortgage    -0.095254   0.216122   0.334950  0.013873  0.165092  0.133382
house_owner -0.075194   0.122856   0.166290  0.174076  0.213718  0.311012
region      -0.002449   0.016457   0.008236  0.006646  0.026636 -0.037631
fam_income   0.110416  -0.310840  -0.393631 -0.086916 -0.241759 -0.179340

                child  occupation  mortgage  house_owner    region  fam_income
gender       0.005021    0.010062 -0.095254    -0.075194 -0.002449    0.110416
education   -0.001692    0.282588  0.216122     0.122856  0.016457   -0.310840
house_val   -0.022602    0.157311  0.334950     0.166290  0.008236   -0.393631
age         -0.254462   -0.231376  0.013873     0.174076  0.006646   -0.086916
online      -0.118371    0.106521  0.165092     0.213718  0.026636   -0.241759
marriage    -0.103286   -0.011196  0.133382     0.311012 -0.037631   -0.179340
child        1.000000    0.090859  0.052138    -0.096108 -0.010958   -0.010293
occupation   0.090859    1.000000  0.186662     0.046050 -0.005777   -0.236152
mortgage     0.052138    0.186662  1.000000     0.259671 -0.034604   -0.345930
house_owner -0.096108    0.046050  0.259671     1.000000 -0.020307   -0.248595
region      -0.010958   -0.005777 -0.034604    -0.020307  1.000000   -0.031313
fam_income  -0.010293   -0.236152 -0.345930    -0.248595 -0.031313    1.000000
```

```
mus_rds = np.random.RandomState(0)                   #correlation matrix heatmap showing the relationships between variables
mus = pd.DataFrame(mus_rds.rand(10, 10))
mus_corr = musei.corr()
mus_corr.style.background_gradient(cmap='YlGnBu').set_precision(2)
#https://stackoverflow.com/questions/29432629/plot-correlation-matrix-using-pandas
```

```
<ipython-input-379-8bd6240eec1d>:4: FutureWarning: this method is deprecated in favour of `Styler.format(precision=..)`
  mus_corr.style.background_gradient(cmap='YlGnBu').set_precision(2)
```

	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_income
gender	1.00	-0.10	-0.08	-0.06	-0.08	-0.02	0.01	0.01	-0.10	-0.08	-0.00	0.11
education	-0.10	1.00	0.24	0.07	0.21	0.05	-0.00	0.28	0.22	0.12	0.02	-0.31
house_val	-0.08	0.24	1.00	0.07	0.13	0.12	-0.02	0.16	0.33	0.17	0.01	-0.39
age	-0.06	0.07	0.07	1.00	0.16	0.17	-0.25	-0.23	0.01	0.17	0.01	-0.09
online	-0.08	0.21	0.13	0.16	1.00	0.17	-0.12	0.11	0.17	0.21	0.03	-0.24
marriage	-0.02	0.05	0.12	0.17	0.17	1.00	-0.10	-0.01	0.13	0.31	-0.04	-0.18
child	0.01	-0.00	-0.02	-0.25	-0.12	-0.10	1.00	0.09	0.05	-0.10	-0.01	-0.01
occupation	0.01	0.28	0.16	-0.23	0.11	-0.01	0.09	1.00	0.19	0.05	-0.01	-0.24
mortgage	-0.10	0.22	0.33	0.01	0.17	0.13	0.05	0.19	1.00	0.26	-0.03	-0.35
house_owner	-0.08	0.12	0.17	0.17	0.21	0.31	-0.10	0.05	0.26	1.00	-0.02	-0.25
region	-0.00	0.02	0.01	0.01	0.03	-0.04	-0.01	-0.01	-0.03	-0.02	1.00	-0.03
fam_income	0.11	-0.31	-0.39	-0.09	-0.24	-0.18	-0.01	-0.24	-0.35	-0.25	-0.03	1.00

Correlation matrix show weak correlation between some variables, while there are strong negative correlation between variables.

Classification

```
#checking if the target variable(flag) is equally distributed)
#As can be seen in the output below, it is equally split therefore no need to downsample or oversample

musei.flag.value_counts()
```

```
Y    20000
N    20000
Name: flag, dtype: int64
```

```
#importing MixMaxScaler from sklearn
#importing StandardScaler from sklearn

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
```

```
 #scaling house_val using StandardScaler

scaler = StandardScaler()
musei[['house_val']]= scaler.fit_transform( musei[['house_val']] )
print (musei)
```

```
      flag  gender  education  house_val  age  online  marriage  child  \
0        Y       0          4   1.064037    1       0         1      2
1        N       1          3  -0.222740    7       0         1      2
2        N       0          2  -0.464383    2       1         1      1
3        Y       0          2   0.111170    2       1         0      2
4        Y       1          2  -0.450314    1       1         1      1
...    ...     ...        ...        ...  ...     ...       ...    ...
39995    Y       1          3  -0.727634    7       1         1      2
39996    N       1          1  -0.221733    4       0         1      2
39997    Y       0          0  -0.410090    3       1         1      2
39998    N       0          1   0.224998    7       1         1      1
39999    N       1          3   1.252498    7       1         1      0

       occupation  mortgage  house_owner  region  fam_income
0               4         1            1       2           2
1               4         1            1       3           7
2               4         1            1       2           4
3               3         1            1       1           2
4               3         1            1       0           6
...           ...       ...          ...     ...         ...
39995           1         1            1       0           8
39996           2         1            1       0          10
39997           3         1            1       2           9
39998           3         1            1       1          12
39999           1         2            1       3           4

[40000 rows x 13 columns]
```

```
#scaling house_val using MinMaxScalerScaler
#MinMaxScaler performed better than StandardScaler
#Therefore MinMaxScaler will be used for scaling house_val
#because it is the only variable which is not in the same dimension

scaler = MinMaxScaler()
musei[['house_val']]= scaler.fit_transform( musei[['house_val']] )
print (musei)
```

```
      flag  gender  education  house_val  age  online  marriage  child  \
0        Y       0          4   0.075646    1       0         1      2
1        N       1          3   0.021317    7       0         1      2
2        N       0          2   0.011115    2       1         1      1
3        Y       0          2   0.035415    2       1         0      2
4        Y       1          2   0.011709    1       1         1      1
...    ...     ...        ...        ...  ...     ...       ...    ...
39995    Y       1          3   0.000000    7       1         1      2
39996    N       1          1   0.021360    4       0         1      2
39997    Y       0          0   0.013407    3       1         1      2
39998    N       0          1   0.040221    7       1         1      1
39999    N       1          3   0.083603    7       1         1      0

       occupation  mortgage  house_owner  region  fam_income
0               4         1            1       2           2
1               4         1            1       3           7
2               4         1            1       2           4
3               3         1            1       1           2
4               3         1            1       0           6
...           ...       ...          ...     ...         ...
```

```
39995          1         1            1       0           8
39996          2         1            1       0          10
39997          3         1            1       2           9
39998          3         1            1       1          12
39999          1         2            1       3           4

[40000 rows x 13 columns]
```

```
musei.head(10)       #Table showing the cleaned version of the first 10 observations for the dataset
```

	flag	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_income
0	Y	0	4	0.075646	1	0	1	2	4	1	1	2	2
1	N	1	3	0.021317	7	0	1	2	4	1	1	3	7
2	N	0	2	0.011115	2	1	1	1	4	1	1	2	4
3	Y	0	2	0.035415	2	1	0	2	3	1	1	1	2
4	Y	1	2	0.011709	1	1	1	1	3	1	1	0	6
5	Y	1	3	0.024869	6	1	1	0	4	2	1	1	7
6	Y	0	3	0.200000	1	1	1	2	4	1	1	3	11
7	N	1	3	0.041693	5	1	1	1	4	1	1	0	5
8	N	1	1	0.020768	4	1	1	1	2	1	0	1	10
9	Y	0	1	0.024138	1	1	1	2	3	1	1	3	7

```
musei.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   flag         40000 non-null  object
 1   gender       40000 non-null  int64
 2   education    40000 non-null  int64
 3   house_val    40000 non-null  float64
 4   age          40000 non-null  int64
 5   online       40000 non-null  int64
 6   marriage     40000 non-null  int64
 7   child        40000 non-null  int64
 8   occupation   40000 non-null  int64
 9   mortgage     40000 non-null  int64
 10  house_owner  40000 non-null  int64
 11  region       40000 non-null  int64
 12  fam_income   40000 non-null  int64
dtypes: float64(1), int64(11), object(1)
memory usage: 4.0+ MB
```

Three classification models will be used for the prediction, namely;

1. Logistics Regression(LR)
2. Random Forest(RF)
3. Support Vector Classifier(SVC)

```
from sklearn.linear_model import LogisticRegression          #Importing logistic regression function from sklearn
from sklearn.ensemble import RandomForestClassifier          #Importing RandomForestClassifier function from sklearn
from sklearn.svm import SVC                                  #Importing Support Vector Classifier function from sklearn

#Functions for splitting data to train/test, including cross validation
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score,precision_score,confusion_matrix
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import StratifiedKFold
```

Logistics Regression

```
#Defining x and y
```

```
x = musei.drop('flag', axis=1)
y = musei.flag

# implementing train-test-split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20)
```

```
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import LabelEncoder

# converting the response variable to numeric.
# Did not convert initially in order to easily identify the class in its categorical state when plotting

le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
```

```
mus_logreg = LogisticRegression()                            # Logistics Regression function
mus_logreg.fit(x_train, y_train)                             # fitting the train data
y_pred = mus_logreg.predict(x_test)                          # Prediction of the LR model

# calculating the accuracy and AUC
LR_accuracy = round(mus_logreg.score(x_train, y_train)* 100,2)
auc_LR = roc_auc_score(y_test, y_pred)

print("LR accuracy:", LR_accuracy)
print("AUC:", auc_LR)
```

```
LR accuracy: 67.63
AUC: 0.6754132264464925
```

```
#Generating confusion matrix for logistics regression

cm_lr = confusion_matrix(y_test, y_pred)
print(cm_lr)
```

```
[[2699 1269]
 [1328 2704]]
```

Random Forest

```
# converting the response variable to numeric.
# Did not convert initially in order to easily identify the class in its categorical state when plotting

le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)

# fit model and make predictions
mus_rf = RandomForestClassifier(n_estimators=100)
mus_rf.fit(x_train, y_train)
Y_pred = mus_rf.predict(x_test)

# calculating the accuracy and AUC
RF_accuracy = round(mus_rf.score(x_train, y_train) * 100, 2)
auc_RF = roc_auc_score(y_test, y_pred)

print("RF accuracy:", RF_accuracy)
print("AUC:", auc_RF)
```

```
RF accuracy: 99.49
AUC: 0.6754132264464925
```

```
#Generating confusion matrix for Random Forest

cm_rf = confusion_matrix(y_test, y_pred)
print(cm_rf)
```

```
[[2699 1269]
 [1328 2704]]
```

Support vector Machine

```
# converting the response variable to numeric.
# Did not convert initially in order to easily identify the class in its categorical state when plotting

le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)


mus_svc = SVC()                                                          # Support Vector Machine function
mus_svc.fit(x_train, y_train)                                            # fitting the train data
y_pred = mus_svc.predict(x_test)                                         # Prediction of SVC model

# calculating the accuracy and AUC
SVC_accuracy = round(mus_svc.score(x_train, y_train) * 100, 2)
auc_SVC = roc_auc_score(y_test, y_pred)

print("SVC accuracy:", SVC_accuracy)
print("AUC:", auc_SVC)
```

```
SVC accuracy: 68.03
AUC: 0.6795614919354839
```

```
#Generating confusion matrix for Support Vector Classifier

cm_svc = confusion_matrix(y_test, y_pred)
print(cm_svc)
```

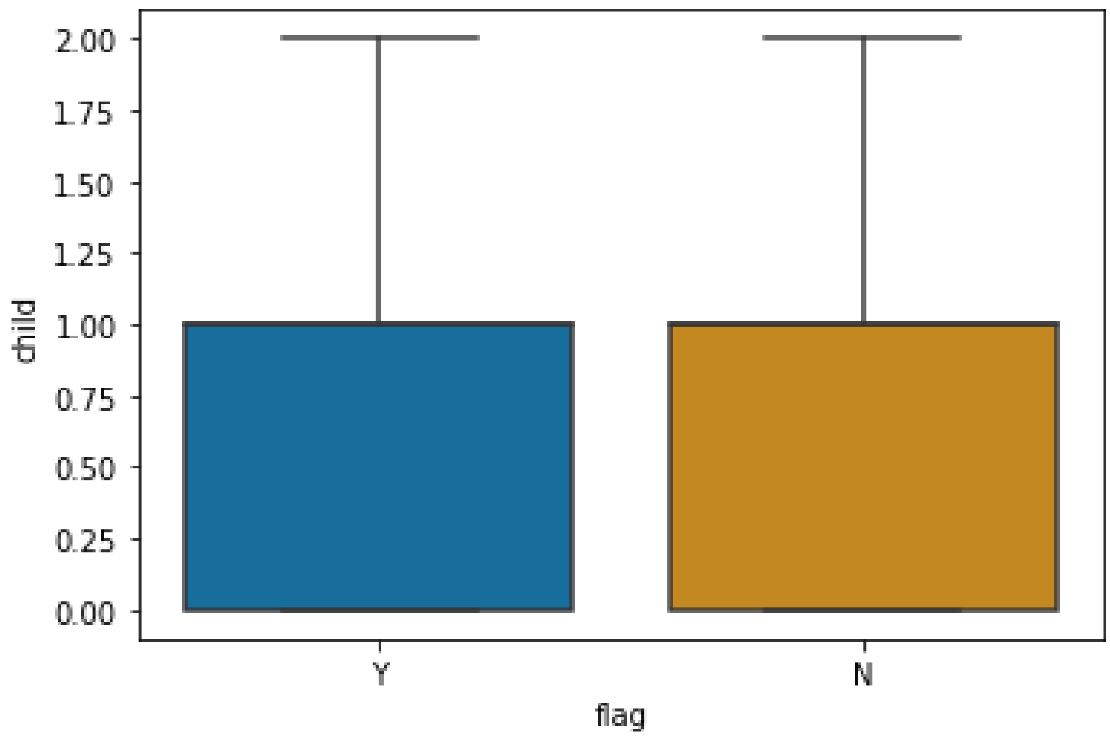
```
[[2665 1303]
 [1260 2772]]
```

✓ 0s completed at 13:20

Appendix B: Other Visualisations

	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_income
gender	1.00	-0.10	-0.08	-0.06	-0.08	-0.02	0.01	0.01	-0.10	-0.08	-0.00	0.11
education	-0.10	1.00	0.24	0.07	0.21	0.05	-0.00	0.28	0.22	0.12	0.02	-0.31
house_val	-0.08	0.24	1.00	0.07	0.13	0.12	-0.02	0.16	0.33	0.17	0.01	-0.39
age	-0.06	0.07	0.07	1.00	0.16	0.17	-0.25	-0.23	0.01	0.17	0.01	-0.09
online	-0.08	0.21	0.13	0.16	1.00	0.17	-0.12	0.11	0.17	0.21	0.03	-0.24
marriage	-0.02	0.05	0.12	0.17	0.17	1.00	-0.10	-0.01	0.13	0.31	-0.04	-0.18
child	0.01	-0.00	-0.02	-0.25	-0.12	-0.10	1.00	0.09	0.05	-0.10	-0.01	-0.01
occupation	0.01	0.28	0.16	-0.23	0.11	-0.01	0.09	1.00	0.19	0.05	-0.01	-0.24
mortgage	-0.10	0.22	0.33	0.01	0.17	0.13	0.05	0.19	1.00	0.26	-0.03	-0.35
house_owner	-0.08	0.12	0.17	0.17	0.21	0.31	-0.10	0.05	0.26	1.00	-0.02	-0.25
region	-0.00	0.02	0.01	0.01	0.03	-0.04	-0.01	-0.01	-0.03	-0.02	1.00	-0.03
fam_income	0.11	-0.31	-0.39	-0.09	-0.24	-0.18	-0.01	-0.24	-0.35	-0.25	-0.03	1.00

Correlation matrix



Appendix C: Table showing unclean and clean dataset

Unclean data during exploration and preprocessing

```
musei.head(10) #exploring the first 10 observation of the dataset
```

	flag	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_income
0	Y	M	4. Grad	756460	1_Unk	N	NaN	U	Professional	1Low	NaN	Midwest	L
1	N	F	3. Bach	213171	7_>65	N	NaN	U	Professional	1Low	Owner	Northeast	G
2	N	M	2. Some College	111147	2_<=25	Y	NaN	Y	Professional	1Low	Owner	Midwest	J
3	Y	M	2. Some College	354151	2_<=25	Y	Single	U	Sales/Service	1Low	NaN	West	L
4	Y	F	2. Some College	117087	1_Unk	Y	Married	Y	Sales/Service	1Low	NaN	South	H
5	Y	F	3. Bach	248694	6_<=65	Y	Married	N	Professional	2Med	Owner	West	G
6	Y	M	3. Bach	2000000	1_Unk	Y	Married	U	Professional	1Low	NaN	Northeast	C
7	N	F	3. Bach	416925	5_<=55	Y	Married	Y	Professional	1Low	Owner	South	I
8	N	F	1. HS	207676	4_<=45	Y	NaN	Y	Blue Collar	1Low	Renter	West	D
9	Y	M	1. HS	241380	1_Unk	Y	Married	U	Sales/Service	1Low	NaN	Northeast	G

Clean data after dealing with data quality issues

```
[ ] musei.head(10)
```

	flag	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_income
0	Y	0	4	0.075646	1	0	1	2	4	1	1	2	2
1	N	1	3	0.021317	7	0	1	2	4	1	1	3	7
2	N	0	2	0.011115	2	1	1	1	4	1	1	2	4
3	Y	0	2	0.035415	2	1	0	2	3	1	1	1	2
4	Y	1	2	0.011709	1	1	1	1	3	1	1	0	6
5	Y	1	3	0.024869	6	1	1	0	4	2	1	1	7
6	Y	0	3	0.200000	1	1	1	2	4	1	1	3	11
7	N	1	3	0.041693	5	1	1	1	4	1	1	0	5
8	N	1	1	0.020768	4	1	1	1	2	1	0	1	10
9	Y	0	1	0.024138	1	1	1	2	3	1	1	3	7

Descriptive Statistics of the dataset

```
musei.describe()
```

	gender	education	house_val	age	online	marriage	child	occupation	mortgage	house_owner	region	fam_inco
ount	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000	40000.0000
nean	0.478300	2.114375	0.030721	4.113800	0.682975	0.872950	0.883050	3.066450	1.387525	0.815225	1.191500	8.1738
std	0.554154	1.189844	0.042221	1.932742	0.465323	0.333033	0.732145	1.115018	0.710501	0.388120	1.156616	2.7332
min	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	1.0000
25%	0.000000	1.000000	0.008066	3.000000	0.000000	1.000000	0.000000	2.000000	1.000000	1.000000	0.000000	7.0000
50%	0.000000	2.000000	0.021487	4.000000	1.000000	1.000000	1.000000	3.000000	1.000000	1.000000	1.000000	9.0000
75%	1.000000	3.000000	0.039376	6.000000	1.000000	1.000000	1.000000	4.000000	2.000000	1.000000	2.000000	10.0000
max	2.000000	4.000000	1.000000	7.000000	1.000000	1.000000	2.000000	5.000000	3.000000	1.000000	4.000000	13.0000