Assignment 1

"PREDICTING CUSTOMER PURCHASE OF LIFE INSURANCE: A MACHINE LEARNING ANALYSIS OF IMPERIALS LTD DATA"

Name: Moon Karmakar

Course Title: Data Mining

Course Code: MGT7216

Student Id: 40389123

Date: 12th march 2023

Word count: 2088

Contents

1. Introduction and Background: -

A vital product that offers people and their family's financial stability is life insurance. Imperials Ltd's marketing division has been keeping track of fourteen different metrics on consumers who purchased their latest life insurance product. They are interested in determining if a potential new client would purchase the product. There are several problems with the data set, such as inconsistent fields, missing values, and incomplete entries(*Machine Learning: A Probabilistic Perspective - Kevin P. Murphy - Google Books*, n.d.)

As it a classification problem, classification defined as the process of predicting class or category from observed values or given data points. According to this issue description, the classified output might take the form of "buyer" or "non-buyer". This study examines data mining and uses the generalised classification model to create a prediction model for whether or not insurance will be purchased. It starts with background information before diving into classification analysis and concentrating on the generalised model (Huang et al., 2011).

The study compares estimation methods for a generalised linear regression model, a nonparametric regression model, and a partial linear model. This article uses simulations to test the model's ability to predict (Lundberg et al., n.d.).

The algorithm can confidently predict whether a customer will purchase insurance. In order to acquire insights into the business challenge, this paper describes the process of cleaning the data and preparing it for analysis, giving descriptive statistics, and creating visualisations.

2. Methodology: -

Classification refers to a predictive modelling problem where a class label is predicted for a given example of input data (Neelamegam et al., n.d.).

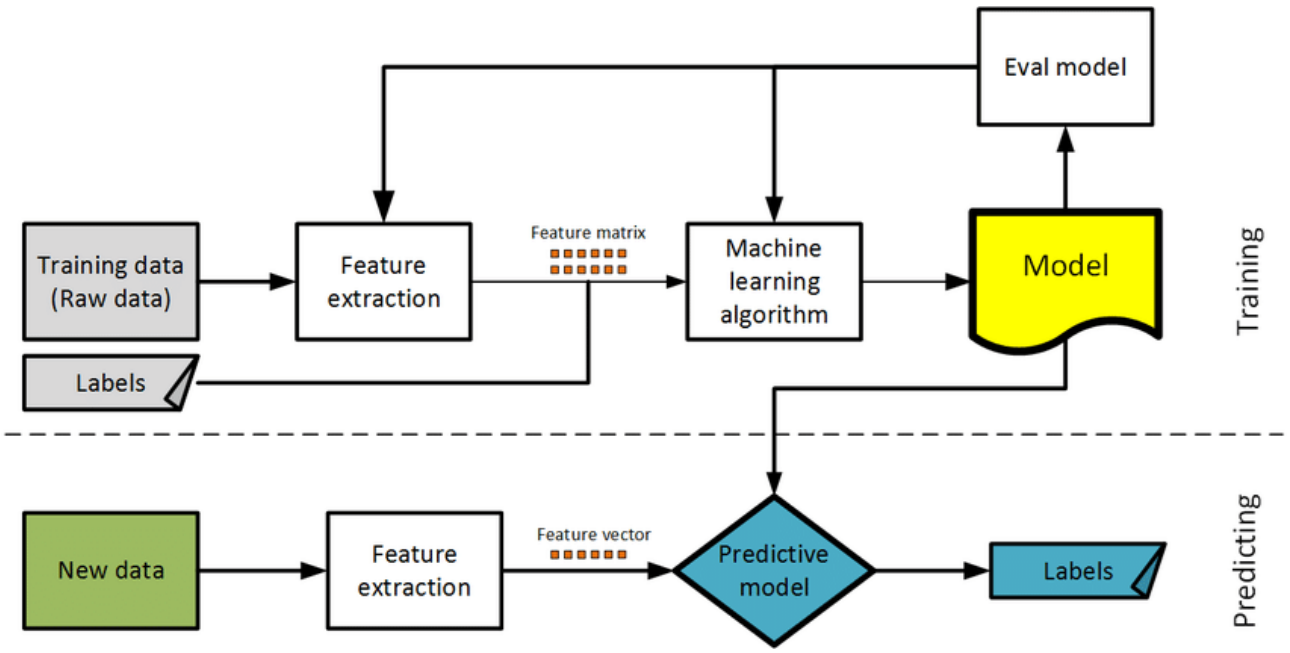
Examples of classification problems include:

1. Given instance, classify if it is buyer or not.
2. Given a handwritten character, classify it as one of the known characters.
3. Given recent user behaviour, classify as churn or not.

From a modelling perspective, classification requires a training dataset with many examples of inputs and outputs from which to learn.

A model will use the training dataset and will calculate how to best map examples of input data to specific class labels. As such, the training dataset must be sufficiently representative of the problem and have many examples of each class label (Brownlee, 2020).

Class labels are often string values, e.g. "0" and "*1*" and must be mapped to numeric values before being provided to an algorithm for modelling. This is often referred to as label encoding, where a unique integer is assigned to each class label, e.g. "non-buyer" = 0, "buyer" = 1(Mendez et al., 2019) .



Picture Source: (Nguyen et al., 2016)

Data Pre-processing

Data pre-processing is a step in the data mining and data analysis process that transforms raw data into a format that computers and machine learning can understand and analyse(Aqajari et al., n.d.).

Machines prefer to process information that is neat and tidy; they read data as 1s and 0s. As a result, calculating structured data such as whole numbers and percentages is simple. Unstructured data, such as text and images, must first be cleaned and formatted before being analysed (Hao & Ho, 2019).

In first look, we saw that we have a very mixed data in our columns like Perhaps different sources use different descriptors for features – for example, *1_Unk* or 2_*Some college*. These value descriptors should all be made uniform(Hastie et al., 2001) .

```
5_<=55    6091
4_<=45    4988
6_<=65    3987
1_Unk     3082
3_<=35    2515
7_>65     2308
2_<=25     587
Name: age, dtype: int64
```

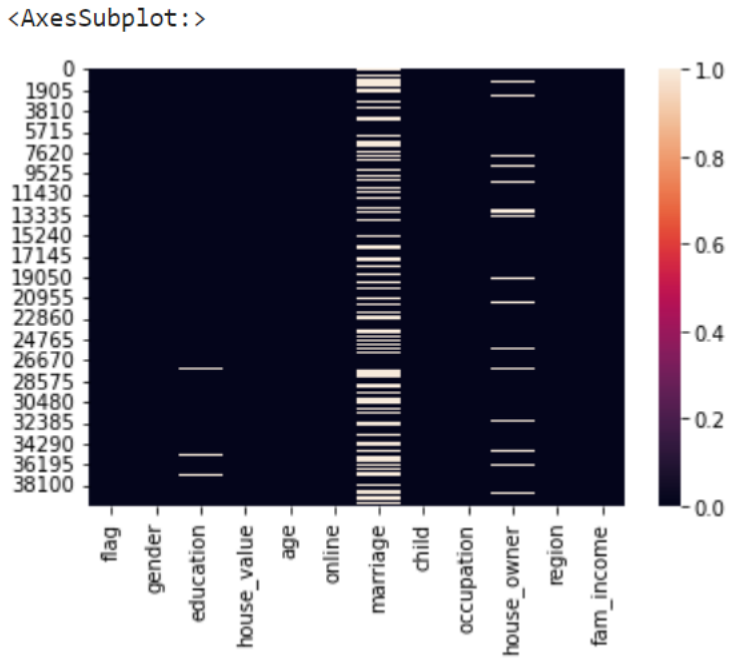
```
#On age
df.replace('1_Unk', '1', inplace=True)
df.replace('2_<=25', '2', inplace=True)
df.replace('3_<=35', '3', inplace=True)
df.replace('4_<=45', '4', inplace=True)
df.replace('5_<=55', '5', inplace=True)
df.replace('6_<=65', '6', inplace=True)
df.replace('7_>65', '7', inplace=True)
```

We replaced some of the columns with their real values; these were mixed-value columns that involved all the columns. Columns are education, age, mortgage, fam_income.

Handling Missing Values: -

We have such some columns where we have null values. One of the biggest impacts of Missing Data is, it can bias the results of the machine learning models or reduce the accuracy of the model. So, it is very important to handle missing values (Saar-Tsechansky & Provost, 2007).

```
flag              0
gender            0
education       741
house_value       0
age               0
online            0
marriage      14027
child             0
occupation        0
house_owner    3377
region            0
fam_income        0
dtype: int64
```



We have two approaches to deal with null values, one is deleting all the null rows or column but it is not preferred because it may occur data loss, or maybe we can lose some of the important relations and second is imputation, it will good to go in many situations, so we used imputation techniques (Zainal Abidin et al., 2018).

Handling Categorical Features: -

From the categorical features we are going to transform the columns education, age, mortgage, and fam_income using label encoding because they have a hierarchy. For the other categories we will treat them as dummy variables (Zhu et al., n.d.).

The next step is to handle categorical data in the dataset after dealing with missing values.

Models are mathematical models that require numbers to function. Categorical data can have multiple values (categories) and be in text form. For instance, Gender: Male/Female/Others, Ranks: 1st/2nd/3rd, and so on (Alalyan et al., n.d.).

Some of the columns are 'education', 'age', 'mortgage', 'fam_income'

```
education ['3. Bach' '2. Some College' '1. HS' '0. <HS' '4. Grad']
age ['7_>65' '2_<=25' '6_<=65' '5_<=55' '4_<=45' '3_<=35']
mortgage ['1Low' '2Med' '3High']
fam_income ['G' 'J' 'L' 'I' 'D' 'E' 'C' 'A' 'F' 'B' 'H' 'K' 'U']
```

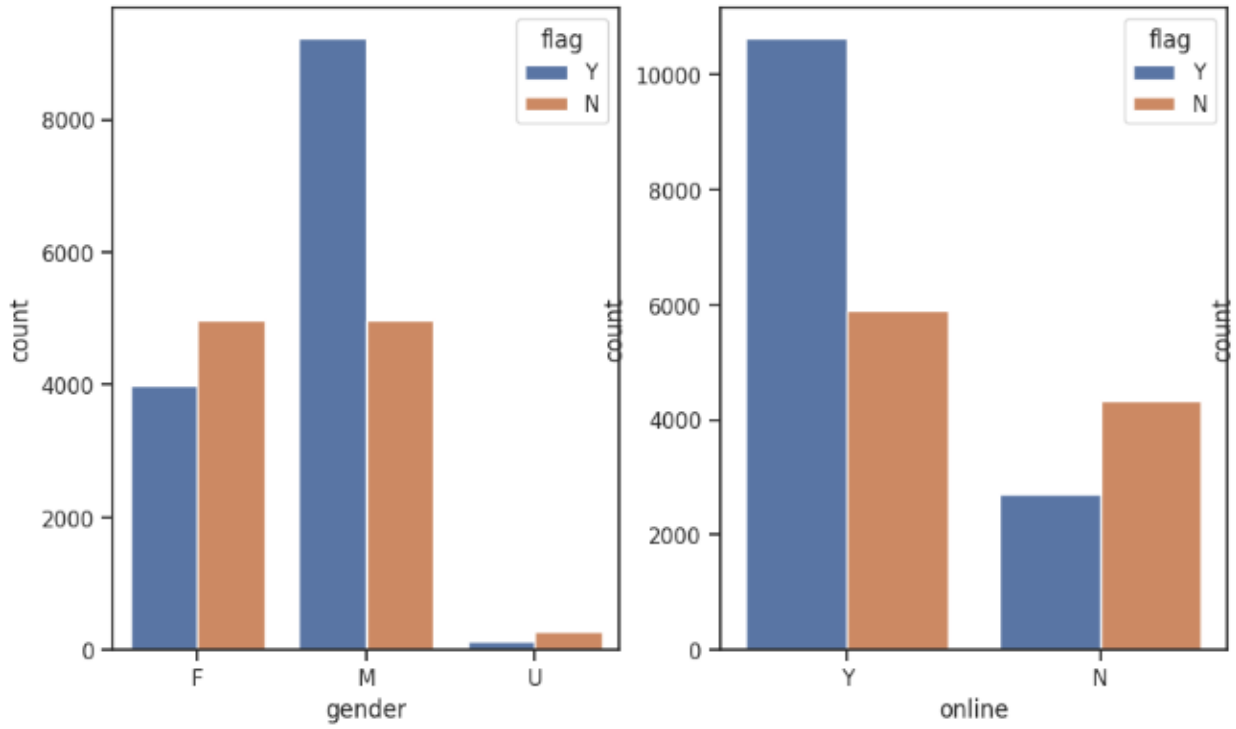
We used Label Encoding Technique. It is a well-known encoding technique for dealing with categorical variables. Based on alphabetical ordering, each label is assigned a unique integer in this technique. Let us look at how to implement label encoding in Python using the scikit-learn library, as well as the challenges associated with it(*Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow - Aurélien Géron - Google Books*, n.d.).

Data Visualisation

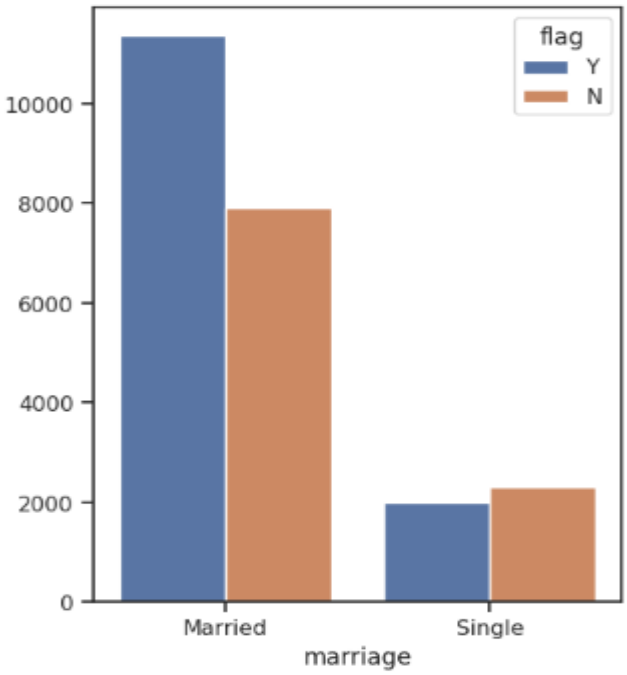
Let us see how different feature are depending on the flag (target) variable.

In first graph we are checking how many customers (buyers and non-buyers) we have according to gender, most of the males purchased the product and unknown gender did not show much interest(Science & 2010, 2010 a).

In second graph we are checking how many customers (buyers and non-buyers) we have who purchase online (Science & 2010, 2010, b).

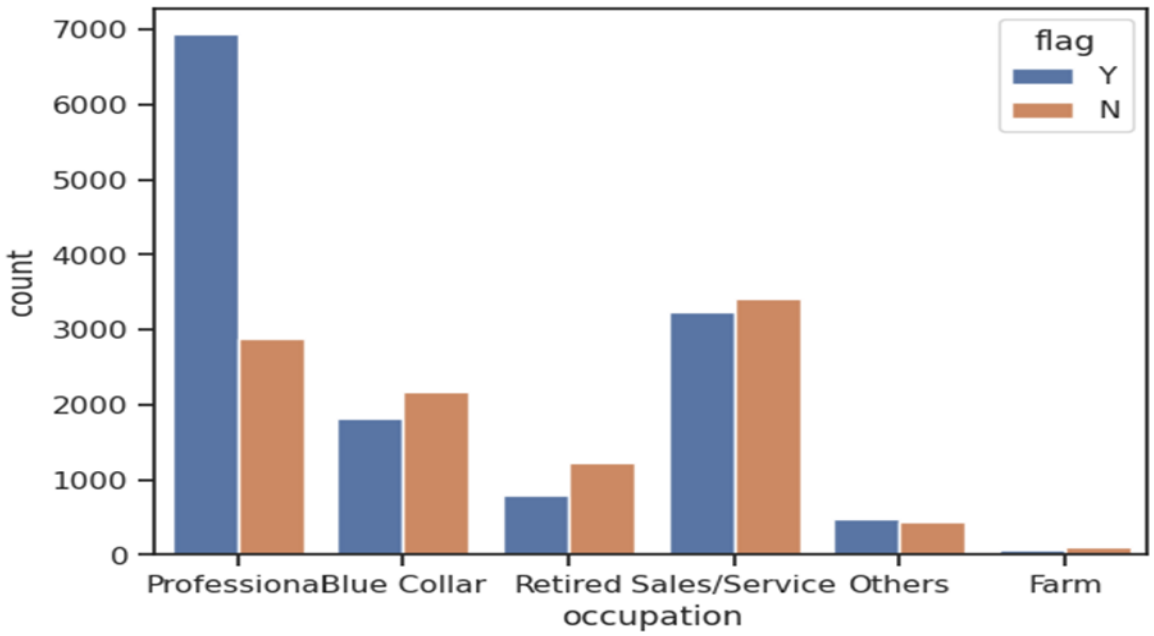


Graph shows that, we have a greater number of insurance buyers those are married and less number of customers those are single.



Visualisation shows that, with increased salary the people are purchasing the product and also we have particular segment like professional, they have more and higher buying rate in comparison to others.

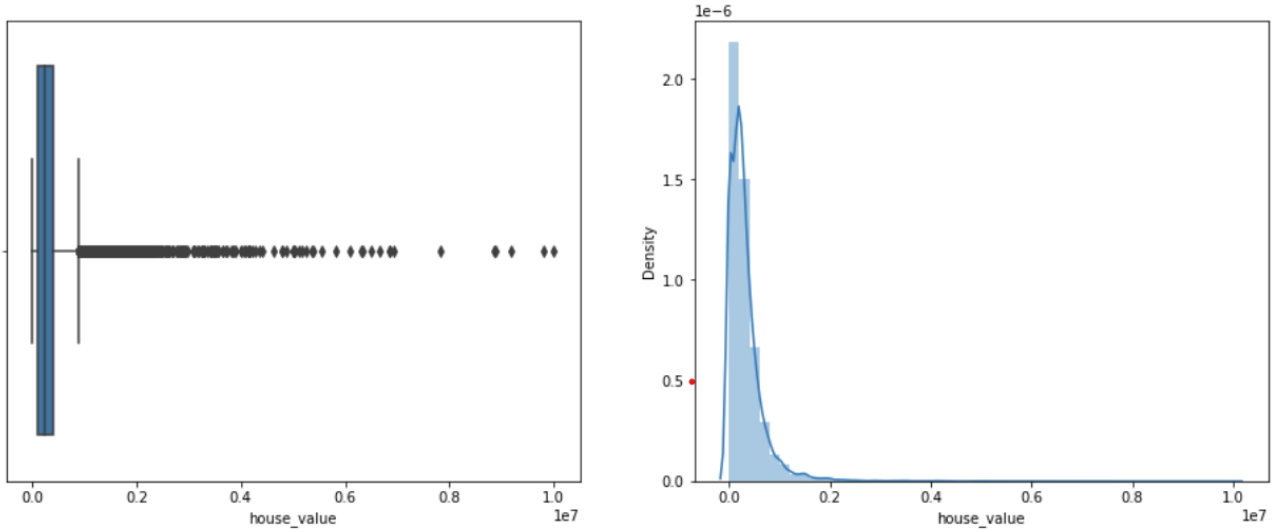
```
<Axes: xlabel='online', ylabel='count'>
```



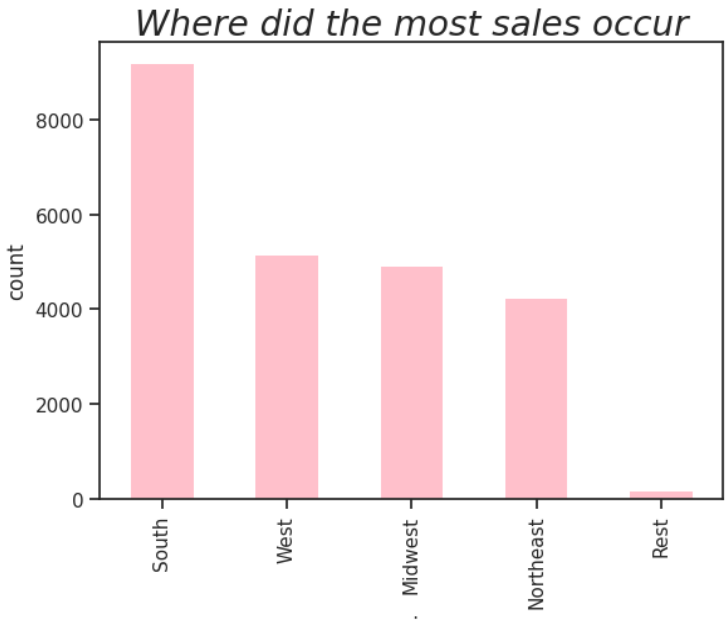
The x-axis in this density map denotes the range of house prices, while the y-axis now shows the density of clients whose houses fall within that range. It is clear from the density map that the distribution of house values is slightly right-skewed.

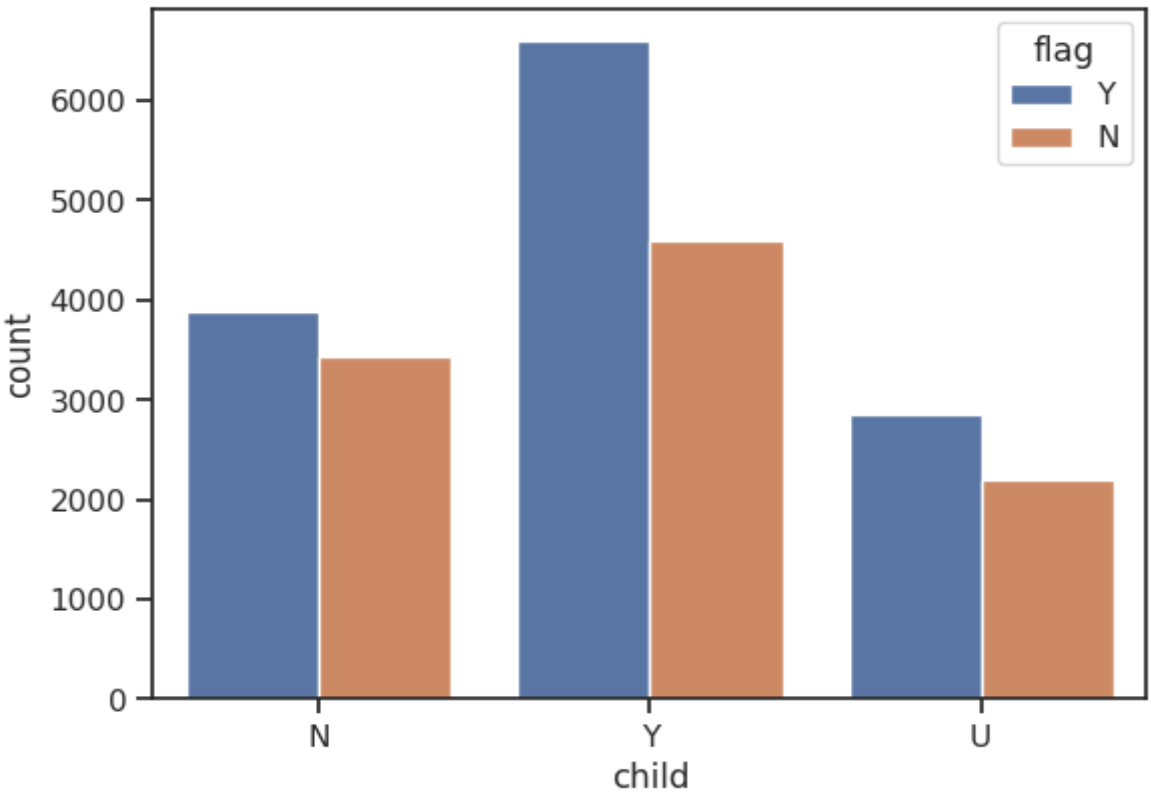
We may better comprehend the range of home prices and the distribution of values within that range by utilising these approaches to visualise the distribution of house values in the dataset.

```
<AxesSubplot:xlabel='house_value', ylabel='Density'>
```

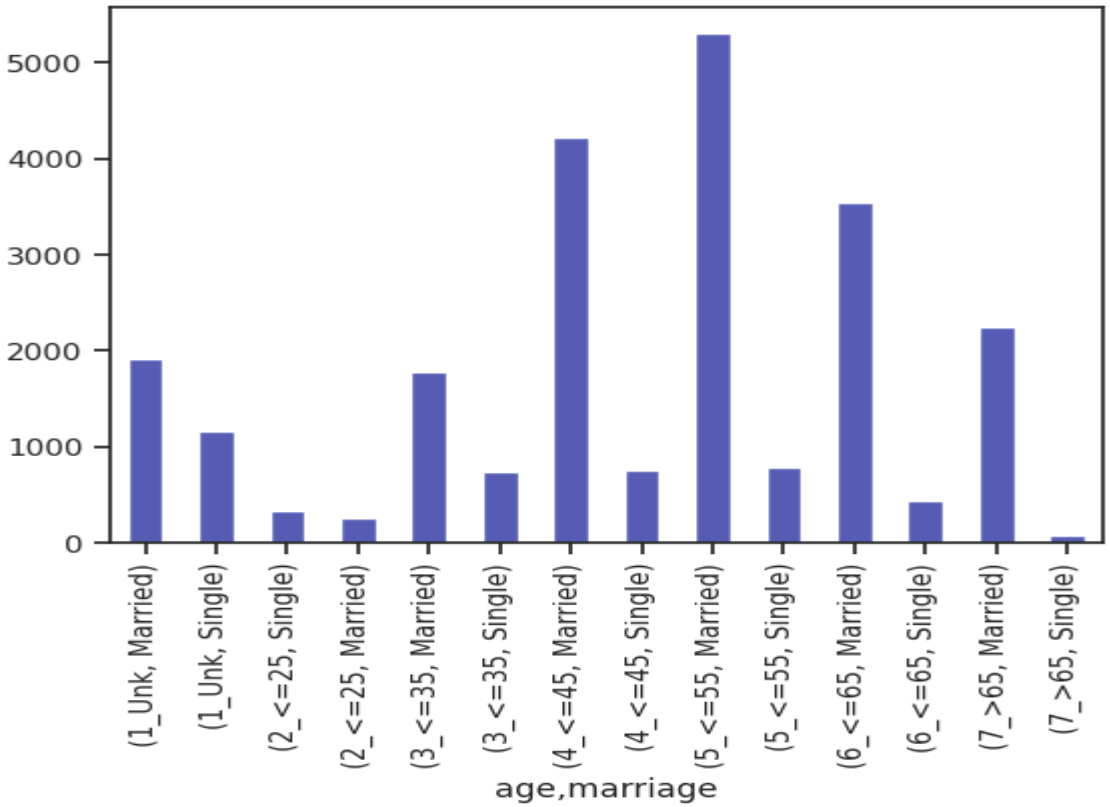


In our dataset we have one demographic column in which we have 5 sites, out of five sites we have large no of customers in south demographic. By area, the proportion of consumers who bought the product is displayed in this visualisation. We found that consumers in the West and South were more inclined to buy the product, but those in the North and East were less likely to do so.

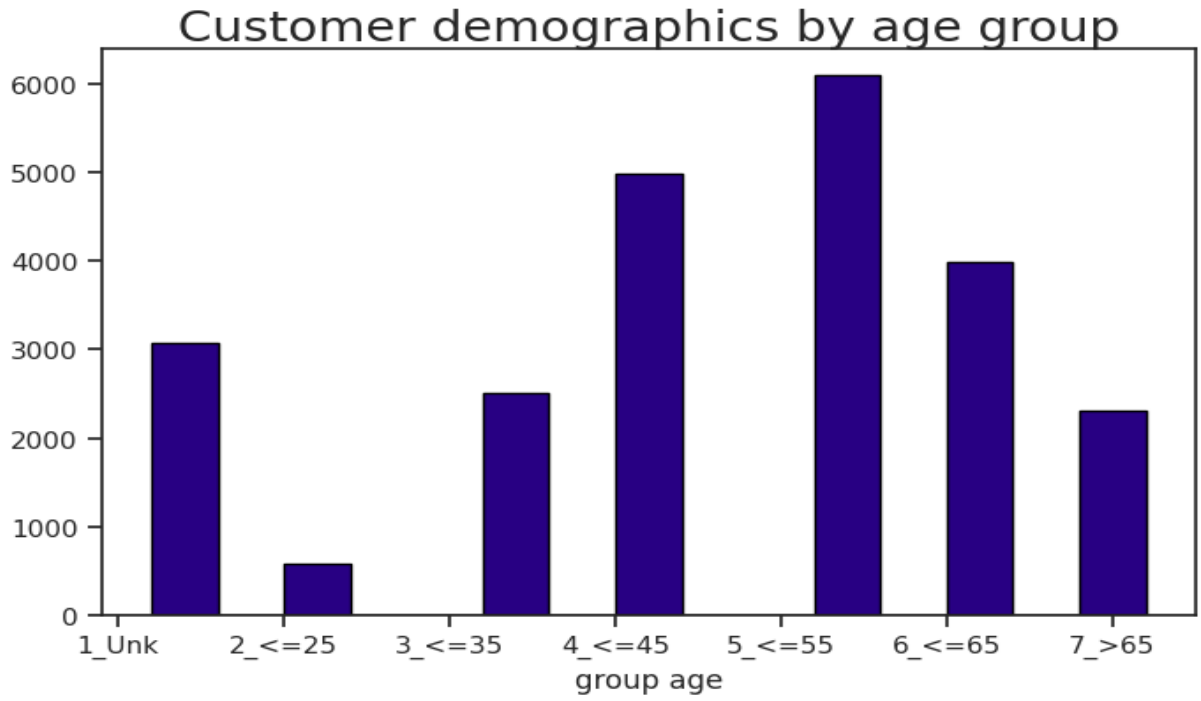




According to the above and below visualisation, we can say that those are married and those have children are more insurance buyers in the company.



As seen by the graph, we are comparing three categories at once: age, marital status, and customers. Since a significant portion of our customers are married and have children, the graph further reinforces the argument made above.



Modelling

Modelling algorithms are developed, trained, and put into use in modelling to simulate logical decision-making based on information at hand. Advanced intelligence approaches including real-time analytics, predictive analytics, and augmented analytics (Society & 1997, 1997).

For our problem statement we have trained three algorithms: -

1. Random Forest
2. Gradient Boosting
3. XGBoost

1. Random Forest: -

A random forest is created by assembling a few decision trees, each of which is trained using a random subset of the data and features. The individual trees are less likely to overfit to the training data and will have various biases and variances because the data and features were chosen at random. By combining all the trees' forecasts, the random forest's ultimate prediction is obtained(Rodriguez-Galiano et al., n.d.).

The key benefits of adopting a random forest are its capacity for handling high-dimensional data, resistance to outliers and noisy data, and capacity for capturing non-linear correlations between

feature and target variable. Random forests can handle categorical and continuous data and are also extremely simple to install. It can be computationally expensive though, and they might not be as easy to understand as other models. They may also be biased if the data is unbalanced or the individual trees are biased (Ao et al., n.d.).

```
rfc_prediction = rfc.predict(X_test)
rfc_score=accuracy_score(y_test, rfc_prediction)
print(rfc_score)
```

```
0.6837423312883436
```

2. Gradient Boosting: -

Gradient Boosting involves adding new decision trees to the model repeatedly while each tree tries to fix the flaws of the preceding trees. In more detail, the technique first fits a single decision tree to the data before sequentially adding more trees to the model, each one attempting to lower the errors of the prior trees (Ye et al., 2009).

Gradient Boosting does this by measuring the gap between the model's predictions and the actual values in the training data using a loss function. The programme then modifies the decision tree's parameters using a gradient descent optimization algorithm to minimise this loss function(statistics & 2001, n.d.).

It can handle both category and numerical data, as well as missing values, which is one of its key advantages. Additionally, it can record intricate nonlinear connections between the features and the desired outcome. Gradient Boosting can, however, be vulnerable to overfitting if the training data are noisy or the model grows very complex. In order to get good performance, it may also be computationally expensive and necessitate careful optimization of its hyperparameters (Pedregosa FABIANPEDREGOSA et al., 2011).

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
gbc = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, random_state=42)
gbc.fit(x_train, y_train)
y_pred = gbc.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

```

```
Accuracy: 0.6952461799660441
```

3. XGBoost: -

XGBoost is like other Gradient Boosting algorithms, but with a few key differences. One significant improvement is that it employs a more advanced regularisation technique known as "shrinkage regularisation," which aids in the prevention of overfitting by reducing the impact of individual trees

on final predictions. Another significant enhancement is that it employs a technique known as "column subsampling," which randomly selects a subset of the features for each tree, reducing the risk of overfitting and speeding up the training process (Chen & He, n.d.).

Because it can handle both sparse and dense input data, XGBoost is particularly well-suited for large-scale, high-dimensional datasets. It also supports a broad range of objective functions and evaluation metrics, making it a versatile tool for a wide range of problems (Chen & He, n.d.).

XGBoost's ability to handle missing data is a notable feature. It accomplishes this by splitting the data in such a way that missing values are always on one side of a split rather than being grouped together. This can lead to improved performance and more precise predictions(Chen et al., 2016) .

Overall, XGBoost is a powerful and adaptable machine learning algorithm that has quickly become a favourite among data scientists and machine learning practitioners.

```
y_pred = xgbmodel.predict(X_test)
y_pred = xgbmodel.predict(X_train)
```

```
print('Test')
print('Precision: {:.2f}% \tRecall: {:.2f}% \t\tF1 Score: {:.2f}%'.format(precision_score
```

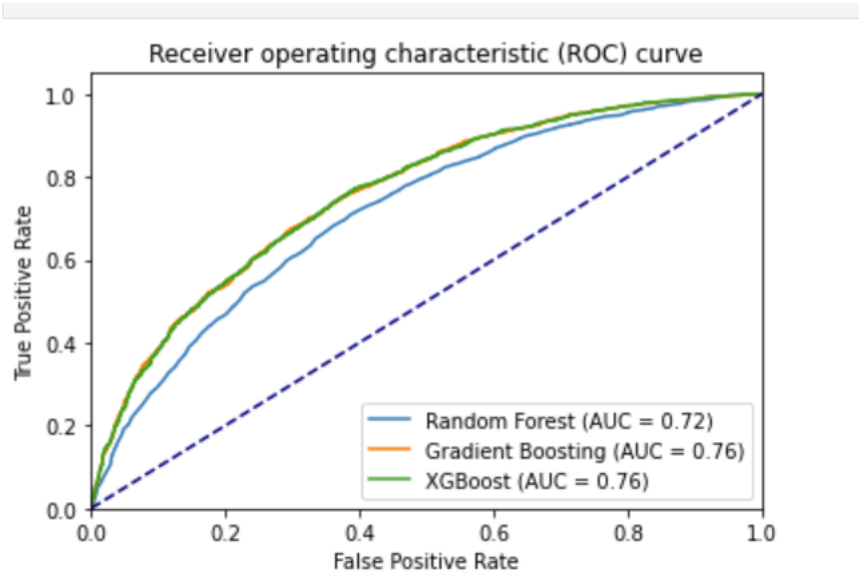
```
Test
Precision: 70.39%       Recall: 71.76%          F1 Score: 71.07%
```

Result: -

Accuracy for all the three models 68%,69%,70% as Random Forest, Gradient Boosting, XGBoost. After checking all other metrics, we are going with XGBoost model for this classification problem.

We train three classification models - XGBoost, Gradient Boosting, and Random Forest on the training data. We predict probabilities on test data using scikit-roc curve learn and auc functions and calculate false positive rate, true positive rate, and AUC-ROC score for each model(Sofaer et al., 2019) .

While AUC-ROC is a useful metric for evaluating binary classification models, it may not be appropriate for multi-class classification problems. In these cases, we can consider using different assessment measures, such as the confusion matrix or the multi-class ROC curve(Saito & Rehmsmeier, 2015).


Receiver operating characteristic (ROC) curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Random Forest (AUC = 0.72)
Gradient Boosting (AUC = 0.76)
XGBoost (AUC = 0.76)

Limitations and Conclusion

In conclusion, we have examined a dataset of clients who purchased the current life insurance policy from Imperials Ltd. Prior to analysis, the dataset was cleaned to address inconsistent fields, bad data formatting, missing values, and incomplete entries. Then, we performed exploratory data analysis, created summary analysis for the dataset, and created several visualisations that shed light on the business issue (Bose & Mahapatra, 2001).

Using three supervised machine learning algorithms—Random Forest, Gradient Boosting, and XGBoost—we trained models to anticipate whether a buyer will buy the life insurance policy. The XGBoost model has the best predicted accuracy of them, with an overall accuracy of 70.39%. Also, we found that major determinants of product purchase were age, marital status, occupation, educational attainment, and income level (Osisanwo et al., 2017).

Based on our data, we discovered that consumers in the West and South areas, as well as those working in executive and administrative positions, were more inclined to buy the goods.

Limited data, model assumptions, overfitting, interpretability, and the complexity of the business environment are a few potential project drawbacks. When evaluating and using the analysis's findings, these limitations need to be taken into account(Bose & Mahapatra, 2001).

Overall, Imperial Ltd may make use of our research to target potential clients and boost sales of the new life insurance product more effectively. The business may concentrate its marketing efforts on those who are most likely to purchase the product by finding key predictors of purchasing behaviour(Boodhun & Jayabalan, 2018).

References

Alalyan, F., … N. Z.-2019 I. 28th, & 2019, undefined. (n.d.). Model-based hierarchical clustering for categorical data. *Ieeexplore.Ieee.Org*. Retrieved March 12, 2023, from https://ieeexplore.ieee.org/abstract/document/8781307/?casa_token=rVYxdoSafhAAAAAA:kWnTXqbFr5gJqQ4eSmAcxjnbh-d0F5Y2Wr5PB1w8otS_J4HH7YVTJYuNNiEcwcWHOTE_RIOC2gQ

Ao, Y., Li, H., Zhu, L., Ali, S., and, Z. Y.-J. of P. S., & 2019, undefined. (n.d.). The linear random forest algorithm and its advantages in machine learning assisted logging regression modeling. *Elsevier*. Retrieved March 12, 2023, from https://www.sciencedirect.com/science/article/pii/S0920410518310635?casa_token=FMPriqX4xQAAAAAA:tad2nR6B57UTz1zHZfZWf9FymjqQNjvtExawMIuo5sEucVWcWIAYQIz4k2KQtskuEajoiRCZ8as

Aqajari, S., Naeini, E., … M. M.-P. C., & 2021, undefined. (n.d.). pyeda: An open-source python toolkit for pre-processing and feature extraction of electrodermal activity. *Elsevier*. Retrieved March 12, 2023, from https://www.sciencedirect.com/science/article/pii/S1877050921006438

Boodhun, N., & Jayabalan, · Manoj. (2018). Risk prediction in life insurance industry using supervised learning algorithms. *Complex & Intelligent Systems 2018 4:2*, *4*(2), 145–154. https://doi.org/10.1007/S40747-018-0072-1

Bose, I., & Mahapatra, R. K. (2001). Business data mining - A machine learning perspective. *Information and Management*, *39*(3), 211–225. https://doi.org/10.1016/S0378-7206(01)00091-X

Brownlee, J. (2020). *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning*. https://books.google.com/books?hl=en&lr=&id=jaXJDwAAQBAJ&oi=fnd&pg=PP1&dq=what+is+classification+in+python&ots=CfNEeGVX3O&sig=Kkuzd-xP_bCquGeB8LVEd6XwSZM

Chen, T., … C. G. on knowledge discovery and data, & 2016, undefined. (2016). Xgboost: A scalable tree boosting system. *Dl.Acm.Org*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785

Chen, T., & He, T. (n.d.). *xgboost: eXtreme Gradient Boosting*.

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow - Aurélien Géron - Google Books. (n.d.). Retrieved March 12, 2023, from https://books.google.co.uk/books?hl=en&lr=&id=X5ySEAAAQBAJ&oi=fnd&pg=PT10&dq=Hands-On+Machine+Learning+with+Scikit-Learn,+Keras,+and+TensorFlow%22+by+Geron,+A.+(2019):+This+book+provides+a+hands-on+approach+to+machine+learning,+including+classification,+using+popular+Python+libraries+such+as+Scikit-Learn,+Keras,+and+TensorFlow.&ots=yB4yve04uK&sig=LGB4ITLBINSHbUSEUPXo_2AWS48&redir_esc=y#v=onepage&q&f=false

Hao, J., & Ho, T. K. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. In *Journal of Educational and Behavioral Statistics* (Vol. 44, Issue 3, pp. 348–361). SAGE Publications Inc. https://doi.org/10.3102/1076998619832248

Hastie, T., Friedman, J., & Tibshirani, R. (2001). *The Elements of Statistical Learning*. https://doi.org/10.1007/978-0-387-21606-5

Huang, L. S., Quaddus, M., Rowe, A. L., & Lai, C. P. (2011). An investigation into the factors affecting knowledge management adoption and practice in the life insurance business. *Knowledge Management Research and Practice*, *9*(1), 58–72. https://doi.org/10.1057/KMRP.2011.2

Lundberg, S., information, S. L.-A. in neural, & 2017, undefined. (n.d.). A unified approach to interpreting model predictions. *Proceedings.Neurips.Cc*. Retrieved March 12, 2023, from https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

Machine Learning: A Probabilistic Perspective - Kevin P. Murphy - Google Books. (n.d.). Retrieved March 12, 2023, from https://books.google.co.uk/books?hl=en&lr=&id=RC43AgAAQBAJ&oi=fnd&pg=PR7&dq=%22Introduction+to+Machine+Learning%22+by+Alpaydin,+E.+(2010):+This+book+provides+a+comprehensive+introduction+to+machine+learning+techniques,+including+classification,+and+their+applications.&ots=ummCfxNtZ5&sig=J79GiKs6YQNUE_gm00TeeQz7SNk&redir_esc=y#v=onepage&q&f=false

Mendez, K. M., Reinke, S. N., & Broadhurst, D. I. (2019). A comparative evaluation of the generalised predictive ability of eight machine learning algorithms across ten clinical metabolomics data sets for binary classification. *Metabolomics*, *15*(12), 1–15. https://doi.org/10.1007/S11306-019-1612-4/FIGURES/5

Neelamegam, S., … E. R.-J. of P. N. T. and, & 2013, undefined. (n.d.). Classification algorithm in data mining: An overview. *Academia.Edu*. Retrieved March 12, 2023, from https://www.academia.edu/download/36131776/type_of_alghoritm.pdf

Nguyen, D., Nguyen, C., Duong-Ba, T., Nguyen, H., Nguyen, A., & Tran, T. (2016). *Joint Network Coding and Machine Learning for Error-prone Wireless Broadcast*. http://arxiv.org/abs/1612.08914

Osisanwo, F., Akinsola, J., … O. A.-… J. of C., & 2017, undefined. (2017). Supervised machine learning algorithms: classification and comparison. *Researchgate.Net*, *48*. https://doi.org/10.14445/22312803/IJCTT-V48P126

Pedregosa FABIANPEDREGOSA, F., Michel, V., Grisel OLIVIERGRISEL, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot andÉdouardand, M., Duchesnay, andÉdouard, & Duchesnay EDOUARDDUCHESNAY, Fré. (2011). Scikit-learn: Machine learning in Python. *Jmlr.Org*, *12*, 2825–2830. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https:/

Rodriguez-Galiano, V., … M. S.-C.-O. G., & 2015, undefined. (n.d.). Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. *Elsevier*. Retrieved March 12, 2023, from https://www.sciencedirect.com/science/article/pii/S0169136815000037?casa_token=Lh1tXwM_GpcAAAAA:0Uy_GOMA8oxKH8oUt0tFEFKP22ouvZx9Qk1-xt5imM3UYUQztBNNS-BQdWj-0umoWLHfPDUkisg

Saar-Tsechansky, M., & Provost, F. (2007). Handling missing values when applying classification models. *Journal of Machine Learning Research*, *8*, 1625–1657. https://www.jmlr.org/papers/volume8/saar-tsechansky07a/saar-tsechansky07a.pdf

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3), e0118432. https://doi.org/10.1371/JOURNAL.PONE.0118432

Science, W. M.-P. of the 9th P. in, & 2010, undefined. (2010). Data structures for statistical computing in python. *Conference.Scipy.Org*. https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf

Society, M. D.-J. of the O. R., & 1997, undefined. (1997). Network and discrete location: models, algorithms and applications. *Taylor & Francis*, *48*(7), 763–764. https://doi.org/10.1057/palgrave.jors.2600828

Sofaer, H. R., Hoeting, J. A., Jarnevich, C. S., & Helen Sofaer, C. R. (2019). The area under the precision-recall curve as a performance metric for rare binary events. *Wiley Online Library*, *10*(4), 565–577. https://doi.org/10.1111/2041-210X.13140

statistics, J. F.-A. of, & 2001, undefined. (n.d.). Greedy function approximation: a gradient boosting machine. *JSTOR*. Retrieved March 12, 2023, from https://www.jstor.org/stable/2699986?casa_token=YDfYMrelmNcAAAAA:Pv9MT6704eAjuHzUo5Nr3sw2hX-ACScHbj4war1NFKHI4-2hpSFGmAHoGP_78KrFVe9llUaL7nLheWgoSAMtKxjJQePUjJMQGznvVsxTGvWteXBM0Ac8

Ye, J., Chow, J. H., Chen, J., & Zheng, Z. (2009). Stochastic gradient boosted distributed decision trees. *International Conference on Information and Knowledge Management, Proceedings*, 2061–2064. https://doi.org/10.1145/1645953.1646301

Zainal Abidin, N., Ritahani Ismail, A., & Emran, N. A. (2018). Performance Analysis of Machine Learning Algorithms for Missing Value Imputation. In *IJACSA) International Journal of Advanced Computer Science and Applications* (Vol. 9, Issue 6). www.ijacsa.thesai.org

Zhu, C., Cao, L., intelligence, J. Y. analysis and machine, & 2020, undefined. (n.d.). Unsupervised heterogeneous coupling learning for categorical representation. *Ieeexplore.Ieee.Org*. Retrieved March 12, 2023, from https://ieeexplore.ieee.org/abstract/document/9145599/?casa_token=U7AVgPE24oMAAAAA:SJcb_iSIISSo6tPlDeH6UGLef6iTUaIeE9LtM8mFOP8E1yZiek_jeNN6nqycjKV4HOG87E35GY0

#Appendix

Load Dataset

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

```
df= pd.read_csv("sales_data.csv")
```

```
df
```

```
df.education.unique()
```

```
array(['4. Grad', '3. Bach', '2. Some College', '1. HS', '0. <HS', nan],
      dtype=object)
```

```
df.occupation.unique()
```

```
array(['Professional', 'Sales/Service', 'Blue Collar', 'Others',
       'Retired', 'Farm'], dtype=object)
```

```
df.age.unique()
```

```
array(['1_Unk', '7_>65', '2_<=25', '6_<=65', '5_<=55', '4_<=45', '3_<=35'],
      dtype=object)
```

```
df.isna().sum()
```

```
null_values=df[df["education"].isnull()]
```

Analysis and Visualization

```
fam_income1= df['fam_income']
fam_income1.value_counts()
```

```
# Age Variable
f,ax=plt.subplots(1,3,figsize=(16,6))
sns.countplot(x='gender',hue='flag',data=df,ax=ax[0])
sns.countplot(x='online',hue='flag',data=df,ax=ax[1])
sns.countplot(x='marriage',hue='flag',data=df,ax=ax[2])
```

```
f,ax=plt.subplots(1,2,figsize=(16,6))
sns.boxplot(x=df["house_value"],ax=ax[0])
sns.distplot(df["house_value"],ax=ax[1])
```

```
#Increasing with the salary the people are purchasing the product..
sns.countplot(x='occupation',hue='flag',data=df)
sns.countplot(x='online',hue='flag',data=df,ax=ax[1])
```

```
print(df['child'].value_counts())
sns.countplot(x='child',hue='flag',data=df)
```

```
#On education
df.replace('0. <HS', 'dropout', inplace=True)
df.replace('1. HS', 'hs', inplace=True)
df.replace('2. Some College', 'associates', inplace=True)
df.replace('3. Bach', 'bachelors', inplace=True)
df.replace('4. Grad', 'masters', inplace=True)

#On age
df.replace('1_Unk', '1', inplace=True)
df.replace('2_<=25', '2', inplace=True)
df.replace('3_<=35', '3', inplace=True)
df.replace('4_<=45', '4', inplace=True)
df.replace('5_<=55', '5', inplace=True)
df.replace('6_<=65', '6', inplace=True)
df.replace('7_>65', '7', inplace=True)

#On mortgage
df.replace('1Low', 'low', inplace=True)
df.replace('2Med', 'medium', inplace=True)
df.replace('3High', 'high', inplace=True)
```

```
x=['M','F','U']
plt.xlabel("gender")
plt.ylabel("count")
plt.title('Customer Predominant Gender', fontsize=20,fontweight=10,  pad='2.0',fontstyle='italic')
Gender.value_counts().plot(kind='bar',color='pink', figsize=(8,6))
plt.show()
```

```
from sklearn.preprocessing import LabelEncoder
#
# Instantiate LabelEncoder
#
le = LabelEncoder()

cols = ['flag','gender', 'education','online', 'marriage', 'child', 'occupation', 'house_owner','region', 'fam_
#
# Encode labels of multiple columns at once
#
df[cols] = df[cols].apply(LabelEncoder().fit_transform)
#
# Print head
#
df.head()
```

```
#Separate data into features(x) and target(y)
x_all = df.drop(['flag'], axis=1)
y_all = df['flag']
```

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x_all, y_all, test_size=0.2, random_state=42)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Train a random forest classifier
rfc = RandomForestClassifier(n_estimators=100, random_state=42)
rfc.fit(x_train, y_train)

# Make predictions on the testing set
y_pred = rfc.predict(x_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.6687181663837012
```

```
gbc = GradientBoostingClassifier(random_state=42)

# Define the parameter grid to search over
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1],
    'max_depth': [3, 5, 7],
    'min_samples_split': [2, 5, 10],
}

# Use GridSearchCV to find the best hyperparameters
grid_search = GridSearchCV(estimator=gbc, param_grid=param_grid, cv=5)
grid_search.fit(x_train, y_train)

# Print the best hyperparameters
print("Best hyperparameters: ", grid_search.best_params_)

# Make predictions on the testing set using the best hyperparameters
best_gbc = grid_search.best_estimator_
y_pred = best_gbc.predict(x_test)

# Evaluate the model using the best hyperparameters
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```python
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
y_pred = dtc.predict(X_test)
```

```python
y_pred_test = dtc.predict(X_test)
y_pred_train = dtc.predict(X_train)
```

```python
print("Accuracy:", accuracy_score(y_true, y_pred))
print("Precision:", precision_score(y_true, y_pred))
print("Recall:", recall_score(y_true, y_pred))
print("F1 Score:", f1_score(y_true, y_pred))
print("AUC-ROC:", roc_auc)
print("Confusion Matrix:\n", conf_mat)
print("Classification Report:\n", class_report)
```