RFM MODEL:-

In [537]:
```
# Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
```

LABELLING TRAINING DATA USING RFM MODEL

Recency, Frequency and Monetary (Profit Margin) are the calculated fields that were created in the Tableau Prep Builder.

- Recency - Is the number of days from the last transaction made by the customer.
- Frequency - Number of Transactions the customer has made during the period of consideration.
- Monetary- Profit amount from all the transactions made by the customer.

In [538]:
```
RFM=pd.read_excel(r"C:\Users\Vicky Yewle\Downloads\KPMG INTERNSHIP\TASK 2\RFM Analysis.xlsx")
RFM.head()
```

Out[538]:

	customer_id	address	postcode	state	country	property_valuation	first_name	last_name	gender	past_3_years_bike_related_purchases	...	deceased_ir
0	2322	496 Summit Road	2120	NSW	Australia	10	Hazlett	Rosenschein	Male	72	...	
1	2278	31953 Dixon Way	2580	NSW	Australia	4	Gerri	Heliet	Male	11	...	
2	1052	06 Declaration Hill	3184	VIC	Australia	11	Dela	Flannigan	Female	31	...	
3	2066	9 Logan Court	2570	NSW	Australia	9	Skipp	McLarens	Male	91	...	
4	2838	50469 Shelley Avenue	4350	QLD	Australia	8	Lydie	Scholfield	Female	14	...	

5 rows × 23 columns

In [539]:
```
RFM.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2519 entries, 0 to 2518
Data columns (total 23 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   customer_id                          2519 non-null   int64  
 1   address                              2519 non-null   object 
 2   postcode                             2519 non-null   int64  
 3   state                                2519 non-null   object 
 4   country                              2519 non-null   object 
 5   property_valuation                   2519 non-null   int64  
 6   first_name                           2519 non-null   object 
 7   last_name                            2443 non-null   object 
 8   gender                               2519 non-null   object 
 9   past_3_years_bike_related_purchases  2519 non-null   int64  
 10  job_title                            2519 non-null   object 
 11  job_industry_category                2519 non-null   object 
 12  wealth_segment                       2519 non-null   object 
 13  deceased_indicator                   2519 non-null   object 
 14  owns_car                             2519 non-null   object 
 15  tenure                               2519 non-null   int64  
 16  count transaction_id                 2519 non-null   int64  
 17  list_price                           2519 non-null   float64
 18  standard_cost                        2519 non-null   float64
 19  Age                                  2519 non-null   int64  
 20  Recency                              2519 non-null   int64  
 21  Frequency                            2519 non-null   int64  
 22  Margin(Monetary)                     2519 non-null   float64
dtypes: float64(3), int64(9), object(11)
memory usage: 452.8+ KB
```

In [540]:
```
col=['customer_id','address','postcode','first_name','last_name','country','deceased_indicator','job_title']
```

In [541]:
```
for cols in col:
    del RFM[cols]
```

In [542]:
```
#hence as we can see that all columns contain equal no of values ie. 2519 non-null values so we can say that there is no NA in
```

Dividing the Dataset into RFM segments:-

Dividing the data into quartiles using the qcut method. Scores are given from 1-4 ranging from Best to Worst respectively.

- Most Recent Customer will get a score of 1.
- Most Frequent customer will get a score of 1.
- The customer who made purchases worth more monetary terms gets a score of 1.
- Below are the quartiles for all the three columns. Have a look at the r_quartile, f_quartile and m_quartile to understand how the scores are being assigned with respect to the quartile values.

In [543]:
```
quartiles = RFM[['Recency','Frequency','Margin(Monetary)']].quantile(q=[0.25,0.50,0.75])
print(quartiles)
```

```
      Recency  Frequency  Margin(Monetary)
0.25     18.0        4.0          1766.355
0.50     45.0        5.0          2785.040
0.75     88.0        7.0          4063.625
```

In [544]:
```
RFM['r_quartile'] = pd.qcut(RFM['Recency'], 4, ['1','2','3','4'])
RFM['f_quartile'] = pd.qcut(RFM['Frequency'], 4, ['4','3','2','1'] )
RFM['m_quartile'] = pd.qcut(RFM['Margin(Monetary)'], 4, ['4','3','2','1'])
```

In [545]:
```
RFM['RFM_Score'] = RFM.r_quartile.astype(str)+ RFM.f_quartile.astype(str) + RFM.m_quartile.astype(str)
RFM.head()
```

Out[545]:

	state	property_valuation	gender	past_3_years_bike_related_purchases	job_industry_category	wealth_segment	owns_car	tenure	count transaction_id	list_price	s
0	NSW	10	Male	72	Health	Mass Customer	Yes	2	4	4035.51	
1	NSW	4	Male	11	Health	Affluent Customer	Yes	8	4	1635.24	
2	VIC	11	Female	31	Financial Services	High Net Worth	No	3	6	8478.41	
3	NSW	9	Male	91	Health	High Net Worth	Yes	12	5	6115.09	
4	QLD	8	Female	14	Retail	High Net Worth	No	21	3	2838.62	

In [546]:
```
#
#4 1
#3 2
#2 3
#1 4

#['111','112','121','211','122','212','221','222','311','312','321','322' ]
```

In [547]:
```
def RFMlabel(x):
    if (x['RFM_Score'] in ['111','112','113','114','121','131','141','122','123','124','132','133','134','142','143','144']):
        return "Gold_customer"

    elif(x['RFM_Score'] in ['211','212','213','214','221','231','241','222','223','224','232','233','234','242','243','244']):
        return "Platinum_customer"

    elif(x['RFM_Score'] in ['311','312','313','314','321','331','341','322','323','324','332','333','334','342','343','344']):
        return "Silver_customer"
    else:
        return "Uranium_customer"
```

In [548]:
```
rfmSeg = RFM
rfmSeg['R_label'] = rfmSeg.apply(RFMlabel,axis=1)
rfmSeg.head()
```

Out[548]:

	state	property_valuation	gender	past_3_years_bike_related_purchases	job_industry_category	wealth_segment	owns_car	tenure	count transaction_id	list_price	s
0	NSW	10	Male	72	Health	Mass Customer	Yes	2	4	4035.51	
1	NSW	4	Male	11	Health	Affluent Customer	Yes	8	4	1635.24	
2	VIC	11	Female	31	Financial Services	High Net Worth	No	3	6	8478.41	
3	NSW	9	Male	91	Health	High Net Worth	Yes	12	5	6115.09	
4	QLD	8	Female	14	Retail	High Net Worth	No	21	3	2838.62	

In [549]:
```
rfmSeg.groupby('R_label').count()
```

Out[549]:

R_label	state	property_valuation	gender	past_3_years_bike_related_purchases	job_industry_category	wealth_segment	owns_car	tenure	cc transactio
Gold_customer	639	639	639	639	639	639	639	639	
Platinum_customer	630	630	630	630	630	630	630	630	
Silver_customer	622	622	622	622	622	622	622	622	
Uranium_customer	628	628	628	628	628	628	628	628	

In [550]:
```
RFM['Age']=pd.cut(RFM.Age,bins=[15,35,50,90],labels=['Young Adults','Middle Age','Older'])
```

In [551]:
```
RFM.head()
```

Out[551]:

	state	property_valuation	gender	past_3_years_bike_related_purchases	job_industry_category	wealth_segment	owns_car	tenure	count transaction_id	list_price	s
0	NSW	10	Male	72	Health	Mass Customer	Yes	2	4	4035.51	
1	NSW	4	Male	11	Health	Affluent Customer	Yes	8	4	1635.24	
2	VIC	11	Female	31	Financial Services	High Net Worth	No	3	6	8478.41	
3	NSW	9	Male	91	Health	High Net Worth	Yes	12	5	6115.09	
4	QLD	8	Female	14	Retail	High Net Worth	No	21	3	2838.62	

In [552]:
```
RFM.groupby('R_label').count()
```

Out[552]:

R_label	state	property_valuation	gender	past_3_years_bike_related_purchases	job_industry_category	wealth_segment	owns_car	tenure	cc transactio
Gold_customer	639	639	639	639	639	639	639	639	
Platinum_customer	630	630	630	630	630	630	630	630	
Silver_customer	622	622	622	622	622	622	622	622	
Uranium_customer	628	628	628	628	628	628	628	628	

In [553]:
```
RFM['Age'] = RFM['Age'].astype(str)
RFM['r_quartile'] = RFM['r_quartile'].astype(str)
RFM['f_quartile'] = RFM['f_quartile'].astype(str)
RFM['m_quartile'] = RFM['m_quartile'].astype(str)
```

In [554]:
```
RFM.dtypes
```

Out[554]:
```
state                                object
property_valuation                    int64
gender                               object
past_3_years_bike_related_purchases   int64
job_industry_category                object
wealth_segment                       object
owns_car                             object
tenure                                int64
count transaction_id                  int64
list_price                          float64
standard_cost                       float64
Age                                  object
Recency                               int64
Frequency                             int64
Margin(Monetary)                    float64
r_quartile                           object
f_quartile                           object
m_quartile                           object
RFM_Score                            object
R_label                              object
dtype: object
```

Data Preprocessing

Let Extract the columns of object datatype and numeric datatype

In [555]:
```
#There seems to be no NA in data however there are certain columns that needs transformation, lets look at all of them
RFM_object = RFM.select_dtypes(include=[object])
RFM_num = RFM.select_dtypes(exclude=[object])
```

In [556]:
```
RFM_object.columns
```

Out[556]:
```
Index(['state', 'gender', 'job_industry_category', 'wealth_segment',
       'owns_car', 'Age', 'r_quartile', 'f_quartile', 'm_quartile',
       'RFM_Score', 'R_label'],
      dtype='object')
```

In [557]:
```
for cols in RFM_object.columns:
    print(RFM_object[cols].value_counts())
    print('------------------------------')
```

```
NSW    1341
VIC     632
QLD     546
Name: state, dtype: int64
------------------------------
Female    1282
Male      1237
Name: gender, dtype: int64
------------------------------
Manufacturing         614
Financial Services    606
Health                479
Retail                258
Property              202
Entertainment         111
IT                    108
Agriculture            87
Telecommunications     54
Name: job_industry_category, dtype: int64
```

In [558]:
```
cols=['state', 'gender', 'job_industry_category', 'wealth_segment',
       'owns_car', 'Age', 'r_quartile', 'f_quartile', 'm_quartile',
       'RFM_Score']
```

In [559]:
```
RFM_dummies = pd.get_dummies(RFM_object, drop_first=True, columns=cols)
RFM_dummies.head()
```

Out[559]:

	R_label	state_QLD	state_VIC	gender_Male	job_industry_category_Entertainment	job_industry_category_Financial Services	job_industry_category_Health	job_
0	Silver_customer	0	0	1	0	0	1	
1	Platinum_customer	0	0	1	0	0	1	
2	Gold_customer	0	1	0	0	1	0	
3	Uranium_customer	0	0	1	0	0	1	
4	Gold_customer	1	0	0	0	0	0	

5 rows × 85 columns

In [560]:
```
cols=['property_valuation', 'past_3_years_bike_related_purchases', 'tenure',
       'count transaction_id', 'list_price', 'standard_cost', 'Recency',
       'Frequency', 'Margin(Monetary)']
```

In [561]:
```
#Scaling Variables
from sklearn.preprocessing import MinMaxScaler as mms

scaler = mms()

RFM_num1= scaler.fit_transform(RFM_num)

```

In [562]:
```
RFM_num2=pd.DataFrame(RFM_num1,
    columns= cols)
```

In [563]:
```
RFM_new= pd.concat([RFM_dummies,RFM_num2], axis=1)
RFM_new
```

Out[563]:

	R_label	state_QLD	state_VIC	gender_Male	job_industry_category_Entertainment	job_industry_category_Financial Services	job_industry_category_Health
0	Silver_customer	0	0	1	0	0	1
1	Platinum_customer	0	0	1	0	0	1
2	Gold_customer	0	1	0	0	1	0
3	Uranium_customer	0	0	1	0	0	1
4	Gold_customer	1	0	0	0	0	0
...	...	...	...	...	...	...	...
2514	Gold_customer	0	1	0	0	1	0
2515	Silver_customer	0	0	1	0	1	0
2516	Gold_customer	0	0	0	0	0	0
2517	Uranium_customer	0	0	0	0	0	0
2518	Uranium_customer	0	0	0	0	0	0

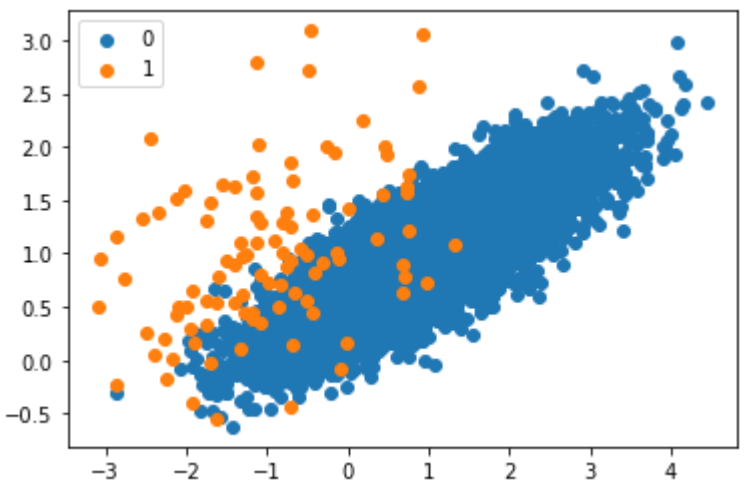
2519 rows × 94 columns

Data Modelling:-

In [564]:
```
# Data Preparation and Model Building

# Importing test_train_split from sklearn library
from sklearn.model_selection import train_test_split

# Putting feature variable to X
X = RFM_new.drop('R_label',axis=1)

# Putting response variable to y
y = RFM_new['R_label']


```

In [565]:
```
# Generate and plot a synthetic imbalanced classification dataset
from collections import Counter
from sklearn.datasets import make_classification
from matplotlib import pyplot
from numpy import where
# define dataset
X, y = make_classification(n_samples=10000, n_features=2, n_redundant=0,
                           n_clusters_per_class=1, weights=[0.99], flip_y=0, random_state=1)
# summarize class distribution
counter = Counter(y)
print(counter)
# scatter plot of examples by class label
for label, _ in counter.items():
    row_ix = where(y == label)[0]
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1], label=str(label))
    pyplot.legend()
pyplot.show()
```

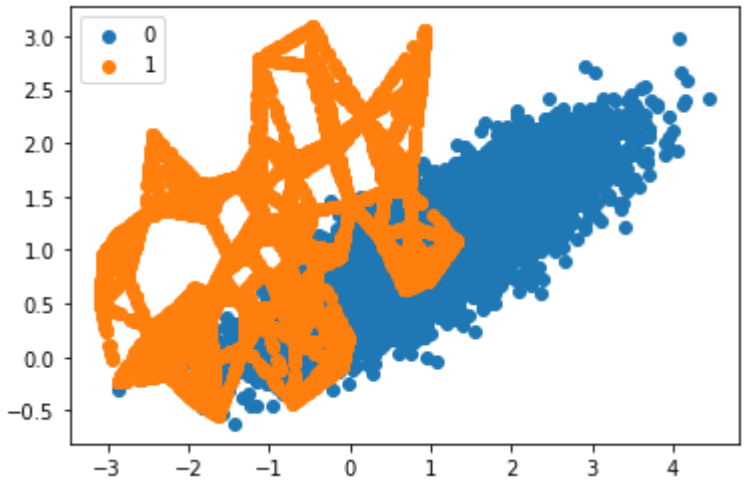
```
Counter({0: 9900, 1: 100})
```



In [566]:
```
# importing SMOTE
from imblearn.over_sampling import SMOTE

# transform the dataset
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
```

In [567]:
```
# Oversample and plot imbalanced dataset with SMOTE
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
from matplotlib import pyplot
from numpy import where
# define dataset
X, y = make_classification(n_samples=10000, n_features=2, n_redundant=0,
n_clusters_per_class=1, weights=[0.99], flip_y=0, random_state=1)
# summarize class distribution
counter = Counter(y)
print(counter)
# transform the dataset
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
# summarize the new class distribution
counter = Counter(y)
print(counter)
# scatter plot of examples by class label
for label, _ in counter.items():
    row_ix = where(y == label)[0]
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1], label=str(label))
pyplot.legend()
pyplot.show()
```

```
Counter({0: 9900, 1: 100})
Counter({0: 9900, 1: 9900})
```



In [568]:
```

from sklearn.model_selection import train_test_split as tts
X_train,X_test,y_train,y_test= tts (X,y,train_size=0.7,test_size=0.3,random_state=100)
```

In [569]:
```
# Importing random forest Regressor from sklearn library
from sklearn.ensemble import RandomForestClassifier

# Running the random forest with default parameters.
rfr = RandomForestClassifier()
```

In [570]:
```
rfr.fit(X_train,y_train)
```

Out[570]:
```
RandomForestClassifier()
```

In [571]:
```
y_pred=rfr.predict(X_test)
```

In [572]:
```
# classification metrics
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[2909  150]
 [ 126 2755]]
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      3059
           1       0.95      0.96      0.95      2881

    accuracy                           0.95      5940
   macro avg       0.95      0.95      0.95      5940
weighted avg       0.95      0.95      0.95      5940
```

In [573]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9535353535353536
```