💻 Question No: 02

⚙️ Setup

- Ensure the Python kernel has the necessary libraries: `pandas` , `seaborn` , `numpy` , `kmeans` , `matplotlib` and `lets-plot`
- Ensure the `Marketing-Customer-Value-Analysis..csv` file is in the `data` folder.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

In [35]:
```
# Load the dataset
df = pd.read_csv("D:/Data Science for Marketing-I& 2/dataset/WA_Fn-UseC_-Marketing-
df.head()
```

Out[35]:

State	Customer Lifetime Value	Response	Coverage	Education	Effective To Date	EmploymentStatus	Gend
ıshington	2763.519279	No	Basic	Bachelor	2/24/11	Employed	
Arizona	6979.535903	No	Extended	Bachelor	1/31/11	Unemployed	
Nevada	12887.431650	No	Premium	Bachelor	2/19/11	Employed	
California	7645.861827	No	Basic	Bachelor	1/20/11	Unemployed	
ıshington	2813.692575	No	Basic	Bachelor	2/3/11	Employed	

ns

i. Create a new column named "Engaged" by transforming the categorical values in the "Response" variable into numerical

values. Why is this transformation important?

In [36]:
```
df['Engaged'] = df['Response'].apply(lambda x: 1 if x == 'Yes' else 0)
```

💡 Convert the categorical "Response" column (values: "Yes" and "No") into numerical values (1 for "Yes" and 0 for "No"),This transformation is important because machine learning models work better with numerical data.

ii. Explain the process of creating dummy variables from the categorical features ['Sales Channel', 'Vehicle Size', 'Vehicle Class', 'Policy', 'Policy Type', 'Employment Status', 'Marital Status', 'Education', 'Coverage'] in the dataset. What function is used, and what is its significance in preparing the data for modeling?

In [37]:
```
continuous_features = [
    'Customer Lifetime Value', 'Income', 'Monthly Premium Auto',
    'Months Since Last Claim', 'Months Since Policy Inception',
    'Number of Open Complaints', 'Number of Policies', 'Total Claim Amount'
]
```

In [38]:
```
columns_to_encode = [
    'Sales Channel', 'Vehicle Size', 'Vehicle Class', 'Policy', 'Policy Type',
    'EmploymentStatus', 'Marital Status', 'Education', 'Coverage'
]
```

In [39]:
```
categorical_features = []
for col in columns_to_encode:
    encoded_df = pd.get_dummies(df[col])
    encoded_df.columns = [col.replace(' ', '.') + '.' + x for x in encoded_df.colum
    categorical_features += list(encoded_df.columns)

    df = pd.concat([df, encoded_df], axis=1)
df.head()
```

Out[39]:

	Customer	State	Customer Lifetime Value	Response	Coverage	Education	Effective To Date	Employ
0	BU79786	Washington	2763.519279	No	Basic	Bachelor	2/24/11	
1	QZ44356	Arizona	6979.535903	No	Extended	Bachelor	1/31/11	
2	AI49188	Nevada	12887.431650	No	Premium	Bachelor	2/19/11	
3	WW63253	California	7645.861827	No	Basic	Bachelor	1/20/11	
4	HB64268	Washington	2813.692575	No	Basic	Bachelor	2/3/11	

5 rows × 66 columns

💡 Categorical data like Sales Channel and Policy Type are converted into numbers using One-Hot Encoding. This allows machine learning models to use them effectively without misinterpreting them as ordered values.

iii. Combine all features into a single dataset for model building.

In [40]:
```
all_features =continuous_features+ categorical_features
response = 'Engaged'
```

In [41]:
```
sample_df = df[all_features + [response]]
sample_df.columns = [x.replace(' ', '.') for x in sample_df.columns]
all_features = [x.replace(' ', '.') for x in all_features]
```

In [42]:
```
sample_df.head()
```

Out[42]:

	Customer.Lifetime.Value	Income	Monthly.Premium.Auto	Months.Since.Last.Claim	Mont
0	2763.519279	56274	69	32	
1	6979.535903	0	94	13	
2	12887.431650	48767	108	18	
3	7645.861827	0	106	18	
4	2813.692575	43836	73	12	

5 rows × 50 columns

In [43]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(sample_df[all_features], sample_
```

In [44]:
```
sample_df.shape
```

Out[44]: (9134, 50)

In [45]:
```
X_train.shape
```

Out[45]: (6393, 49)

In [46]:
```
X_test.shape
```

Out[46]: (2741, 49)

💡 merge numerical and encoded categorical features into a single dataset. This ensures the model has all relevant information needed for predicting engagement.

iv. Split the dataset into a 70:30 ratio for training and test data, and build a Random Forest model using the training data.

In [47]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [48]:
```
rf_model = RandomForestClassifier(
    n_estimators=200,#tree depth(number of estimators)
    max_depth=5
)
```

In [49]:
```
rf_model.fit(X=X_train, y=y_train)
```

Out[49]:
```
RandomForestClassifier(max_depth=5, n_estimators=200)
```

In [50]:
```
#rf_model.estimators_
```

In [51]:
```
rf_model.estimators_[0]
```

Out[51]:
```
DecisionTreeClassifier(max_depth=5, max_features='sqrt', random_state=1962
21789)
```

In [52]:
```
rf_model.predict(X_test)[:23]#
```

Out[52]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0])
```

💡 The dataset is split into 70% training and 30% testing, and a Random Forest model with 200 trees is trained. This helps in understanding customer engagement patterns while preventing overfitting.

v. After fitting the Random Forest model, how can you interpret the output of rf_model.feature_importances_? What does this tell you about the features in the dataset?

In [53]:
```
rf_model.feature_importances_
```

Out[53]:
```
array([0.06320528, 0.08139527, 0.05073066, 0.03250625, 0.05105034,
       0.0115258 , 0.02126309, 0.07021678, 0.041355  , 0.00723882,
       0.01086695, 0.00421635, 0.00797797, 0.007008  , 0.0092317 ,
       0.00453347, 0.0012738 , 0.00305088, 0.00410957, 0.00380892,
       0.00336894, 0.00089179, 0.00063008, 0.0019675 , 0.00205901,
       0.00092485, 0.00133044, 0.0008458 , 0.00031317, 0.00118375,
       0.00085377, 0.00069391, 0.00125017, 0.0076124 , 0.03034657,
       0.00483243, 0.30461412, 0.0230599 , 0.05043076, 0.02136478,
       0.01571811, 0.00515186, 0.00439865, 0.00715945, 0.00493307,
       0.00630607, 0.0044889 , 0.00435221, 0.00235263])
```

In [54]:
```
feature_importance_df = pd.DataFrame(list(zip(rf_model.feature_importances_, all_fe
feature_importance_df.columns = ['feature.importance', 'feature']

feature_importance_df.sort_values(by='feature.importance', ascending=False)#zip joi
```

Out[54]:

	feature.importance	feature
36	0.304614	EmploymentStatus.Retired
1	0.081395	Income
7	0.070217	Total.Claim.Amount
0	0.063205	Customer.Lifetime.Value
4	0.051050	Months.Since.Policy.Inception
2	0.050731	Monthly.Premium.Auto
38	0.050431	Marital.Status.Divorced
8	0.041355	Sales.Channel.Agent
3	0.032506	Months.Since.Last.Claim
34	0.030347	EmploymentStatus.Employed
37	0.023060	EmploymentStatus.Unemployed
39	0.021365	Marital.Status.Married
6	0.021263	Number.of.Policies
40	0.015718	Marital.Status.Single
5	0.011526	Number.of.Open.Complaints
10	0.010867	Sales.Channel.Call.Center
14	0.009232	Vehicle.Size.Small
12	0.007978	Vehicle.Size.Large
33	0.007612	EmploymentStatus.Disabled
9	0.007239	Sales.Channel.Branch
43	0.007159	Education.Doctor
13	0.007008	Vehicle.Size.Medsize
45	0.006306	Education.Master
41	0.005152	Education.Bachelor
44	0.004933	Education.High.School.or.Below
35	0.004832	EmploymentStatus.Medical.Leave
15	0.004533	Vehicle.Class.Four-Door.Car
46	0.004489	Coverage.Basic
42	0.004399	Education.College
47	0.004352	Coverage.Extended

	feature.importance	feature
11	0.004216	Sales.Channel.Web
18	0.004110	Vehicle.Class.SUV
19	0.003809	Vehicle.Class.Sports.Car
20	0.003369	Vehicle.Class.Two-Door.Car
17	0.003051	Vehicle.Class.Luxury.SUV
48	0.002353	Coverage.Premium
24	0.002059	Policy.Personal.L1
23	0.001968	Policy.Corporate.L3
26	0.001330	Policy.Personal.L3
16	0.001274	Vehicle.Class.Luxury.Car
32	0.001250	Policy.Type.Special.Auto
29	0.001184	Policy.Special.L3
25	0.000925	Policy.Personal.L2
21	0.000892	Policy.Corporate.L1
30	0.000854	Policy.Type.Corporate.Auto
27	0.000846	Policy.Special.L1
31	0.000694	Policy.Type.Personal.Auto
22	0.000630	Policy.Corporate.L2
28	0.000313	Policy.Special.L2

💡 The feature importance scores show which factors influence engagement the most. High importance means that feature plays a key role in predicting customer behavior

vi. Discuss how you would evaluate the performance of the Random Forest model after making predictions on the training and test set. Use all possible metrics, and explain why they are important for assessing model effectiveness.

In [55]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score
```

In [56]:
```
in_sample_preds = rf_model.predict(X_train)
out_sample_preds = rf_model.predict(X_test)
```

In [57]:
```
print('In-Sample Accuracy: %0.4f' % accuracy_score(y_train, in_sample_preds)) #insa
print('Out-of-Sample Accuracy: %0.4f' % accuracy_score(y_test, out_sample_preds))
```

```
In-Sample Accuracy: 0.8800
Out-of-Sample Accuracy: 0.8679
```

In [58]:
```
print('In-Sample Precision: %0.4f' % precision_score(y_train, in_sample_preds))
print('Out-of-Sample Precision: %0.4f' % precision_score(y_test, out_sample_preds))
```

```
In-Sample Precision: 0.9853
Out-of-Sample Precision: 0.9123
```

In [59]:
```
print('In-Sample Recall: %0.4f' % recall_score(y_train, in_sample_preds))
print('Out-of-Sample Recall: %0.4f' % recall_score(y_test, out_sample_preds))
```

```
In-Sample Recall: 0.1491
Out-of-Sample Recall: 0.1271
```

The model achieves high accuracy (~87%) but has low recall, meaning it struggles to detect engaged customers. Improving recall with different thresholds or class balancing may be necessary.

- ROC & AUC

In [60]:
```
from sklearn.metrics import roc_curve, auc
```

In [61]:
```
in_sample_preds = rf_model.predict_proba(X_train)[:,1]
out_sample_preds = rf_model.predict_proba(X_test)[:,1]
```

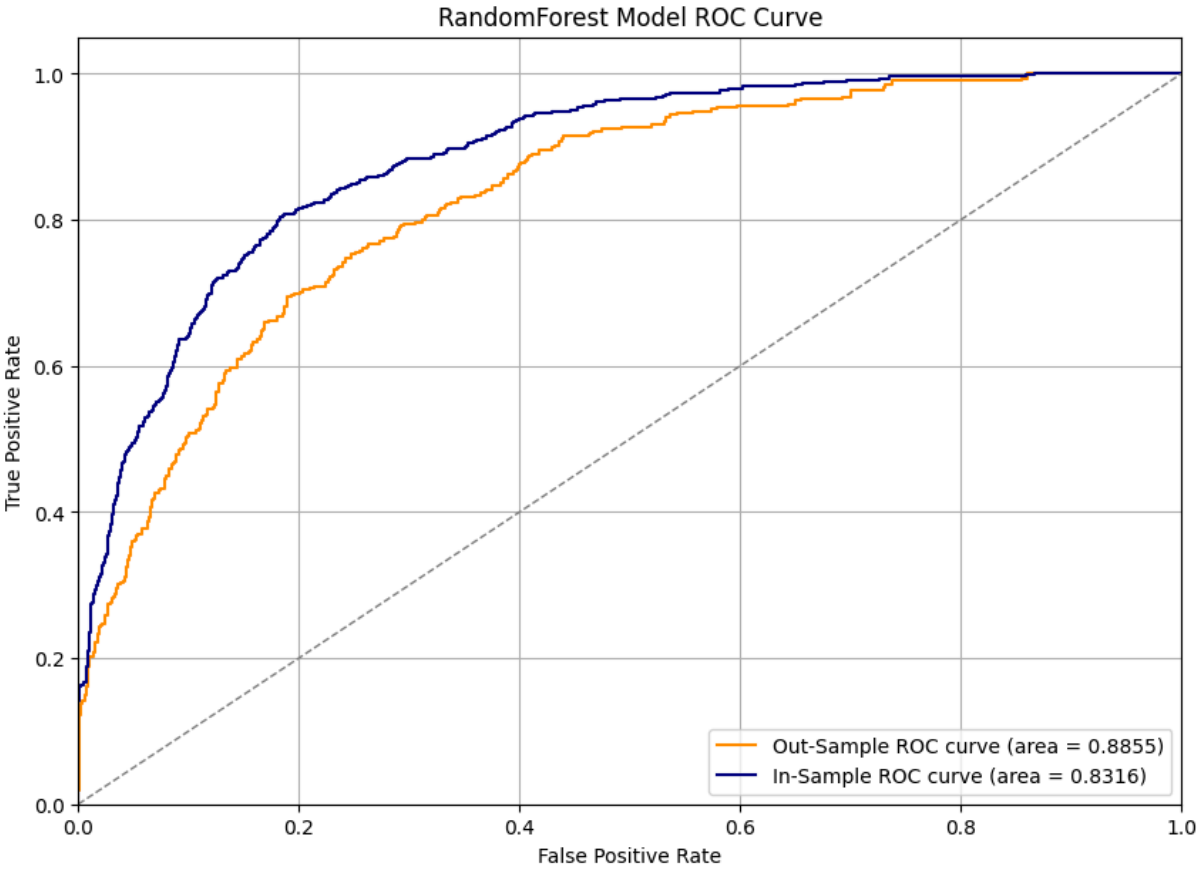
In [62]:
```
in_sample_fpr, in_sample_tpr, in_sample_thresholds = roc_curve(y_train, in_sample_p
out_sample_fpr, out_sample_tpr, out_sample_thresholds = roc_curve(y_test, out_sampl
```

In [63]:
```
in_sample_roc_auc = auc(in_sample_fpr, in_sample_tpr)
out_sample_roc_auc = auc(out_sample_fpr, out_sample_tpr)

print('In-Sample AUC: %0.4f' % in_sample_roc_auc)
print('Out-Sample AUC: %0.4f' % out_sample_roc_auc)
```

```
In-Sample AUC: 0.8855
Out-Sample AUC: 0.8316
```

In [64]:
```
plt.figure(figsize=(10,7))

plt.plot(
    out_sample_fpr, out_sample_tpr, color='darkorange', label='Out-Sample ROC curve
)
plt.plot(
    in_sample_fpr, in_sample_tpr, color='navy', label='In-Sample ROC curve (area =
)
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')
plt.grid()
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('RandomForest Model ROC Curve')
plt.legend(loc="lower right")
```

```
plt.show()
```



💡 The RandomForest model is performing well, with a good balance between in-sample and out-sample performance. The slightly higher out-sample AUC could be due to randomness in data splits or an indication that the test set has more easily separable data. Further evaluation (e.g., cross-validation, feature importance analysis) may be useful to confirm stability.