Experiment No.11

Aim: To build a Business Intelligence Mini Project.

Insurance Lead Prediction(dataset):

For the data and objective, it is evident that this is a Binary Classification Problem data in the Tabular Data format. A policy is recommended to a person when they land on an insurance website, and if the person chooses to fill up a form to apply, it is considered a Positive outcome (Classified as lead). All other conditions are considered Zero outcomes.

It contains attributes namely ID, City_Code, Region_Code, Accomodation_Type, Reco_Insurance_Type, Upper_Age, Lower_Age, Is_Spouse, Health Indicator, Holding_Policy_Duration, Holding_Policy_Type, Reco_policy_Cat, Reco_policy_Premium, Response.
Accomodation_Type, Is_Spouse, Reco_Insurance_Type are categorical attributes while other attributes are numerical attributes. The target attribute that is Response is a binary attribute i.e. 0 / 1.
Dataset link: Insurance Lead Prediction Dataset

Problem Statement:

Insurance companies rely heavily on acquiring new customers to increase their revenue and grow their business. However, identifying potential customers who are likely to purchase insurance policies can be a challenging task. It requires a deep understanding of customer behavior, preferences, and demographics.

In order to address this challenge, the problem statement is to develop a predictive model that can identify potential customers who are likely to purchase insurance policies. The predictive model will help the insurance company to improve their sales performance by targeting leads that are most likely to convert. This will increase the efficiency of the sales process and ultimately lead to increased revenue.

The predictive model will leverage various data sources, such as demographic information, economical condition, and customer behavior, to make accurate predictions about which leads have a higher probability of converting. By identifying the best leads, the insurance company can focus their resources on those leads to improve conversion rates and overall sales performance.

The development of an effective predictive model is crucial for insurance companies to remain competitive in the market. By leveraging business intelligence tools and techniques, the insurance company can gain insights into customer behavior, preferences, and demographics, and develop targeted marketing strategies to increase their customer base and revenue.

Data Mining Task:

The data mining task required for the insurance lead prediction project would be a supervised classification task. This involves building a model that can predict the likelihood of a lead converting to a customer based on various attributes such as demographic information, previous purchase history, and customer behavior. The model would be trained on a dataset consisting of labeled instances, where the label indicates whether a lead converted or not.

The goal of the supervised classification task in this project is to develop a predictive model that can accurately identify potential customers who are likely to purchase insurance policies. The model would take into account various factors such as age, gender, income, occupation, and output the probability of a lead converting to a customer.

The performance of the model would be evaluated using metrics such as accuracy, precision, recall, and F1 score. The model with the highest performance would be selected and used to develop a data-driven strategy for the insurance company to target the best leads and improve their sales performance.

In summary, the data mining task required for the insurance lead prediction project is a supervised classification task, where the goal is to predict the likelihood of a lead converting to a customer based on various attributes. The development of an accurate predictive model can help insurance companies to improve their sales performance by targeting the best leads and ultimately increasing their revenue.

Preprocessing -

For this problem, we can use the Insurance Lead Prediction Dataset available on Kaggle. This dataset includes information about customers with its details along with its response. The dataset contains 50882 records and 14 attributes. The detailed information about the dataset is as follows -

No.	Column Name	Data Type
1	ID	int64
2	City_Code	object
3	Region_Code	int64
4	Accomodation_Type	object
5	Reco_Insurance_Type	object
6	Upper_Age	int64
7	Lower_Age	int64
8	Is_Spouse	object
9	Health Indicator	object
10	Holding_Policy_Duration	object
11	Holding_Policy_Type	float64
12	Reco_Policy_Cat	int64
13	Reco_Policy_Premium	float64
14	Response	int64

```
df.describe()
```

	Region_Code	Upper_Age	Lower_Age	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response
count	50882.000000	50882.000000	50882.000000	30631.000000	50882.000000	50882.000000	50882.000000
mean	1732.788707	44.856275	42.738866	2.439228	15.115188	14183.950069	0.239947
std	1424.081652	17.310271	17.319375	1.025923	6.340663	6590.074873	0.427055
min	1.000000	18.000000	16.000000	1.000000	1.000000	2280.000000	0.000000
25%	523.000000	28.000000	27.000000	1.000000	12.000000	9248.000000	0.000000
50%	1391.000000	44.000000	40.000000	3.000000	17.000000	13178.000000	0.000000
75%	2667.000000	59.000000	57.000000	3.000000	20.000000	18096.000000	0.000000
max	6194.000000	75.000000	75.000000	4.000000	22.000000	43350.400000	1.000000

```
df.isnull().sum().count
```

```
<bound method Series.count of City_Code
Region_Code                    0
Accomodation_Type              0
Reco_Insurance_Type            0
Upper_Age                      0
Lower_Age                      0
Is_Spouse                      0
Health Indicator           11691
Holding_Policy_Duration    20251
Holding_Policy_Type        20251
Reco_Policy_Cat                0
Reco_Policy_Premium            0
Response                       0
dtype: int64>
```

```
[24] df['Holding_Policy_Type'].mode()
```

```
0    3.0
Name: Holding_Policy_Type, dtype: float64
```

```
[25] df['Holding_Policy_Type'].fillna(df['Holding_Policy_Type'].mode()[0], inplace=True)
```

```
[26] df.dropna(inplace=True)
```

```
[27] df.isnull().sum()
```

```
City_Code                  0
Region_Code                0
Accomodation_Type          0
Reco_Insurance_Type        0
Upper_Age                  0
Lower_Age                  0
Is_Spouse                  0
Health Indicator           0
Holding_Policy_Duration    0
Holding_Policy_Type        0
Reco_Policy_Cat            0
Reco_Policy_Premium        0
Response                   0
dtype: int64
```

```
[28] df = pd.get_dummies(df, columns=['Accomodation_Type','City_Code','Reco_Insurance_Type','Is_Spouse','Health Indicator','Holding_Policy_Duration'])
     df.head()
```

ID	Region_Code	Upper_Age	Lower_Age	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response	Accomodation_Type_Owned	Accomodation_Type_Rented	City_Code_C1
1	3213	36	36	3.0	22	11628.0	0	0	1	0
4	4378	52	48	3.0	19	17780.0	0	1	0	0
5	2190	44	44	1.0	16	10404.0	0	0	1	0
6	1785	52	52	1.0	22	15264.0	1	0	1	0
8	3175	75	73	4.0	17	29344.0	1	1	0	1

5 rows × 73 columns

```
[29] df['Reco_Policy_Premium'] = ((df['Reco_Policy_Premium'] - df['Reco_Policy_Premium'].mean())/df['Reco_Policy_Premium'].std())
```

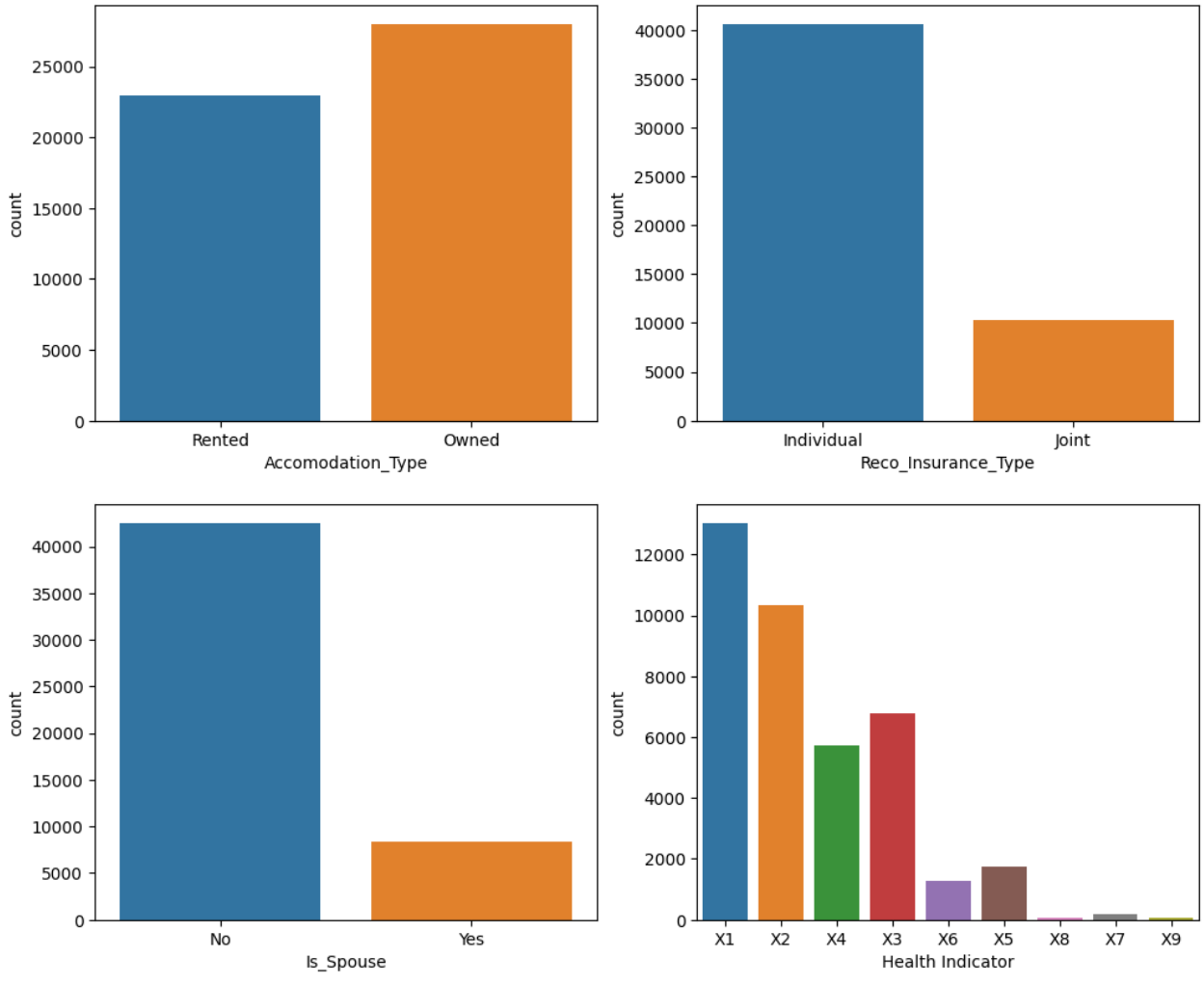
EDA -

These four are count plots each showing the number of entries that have the same values. Accommodation_Type is of two types either Rented or Owned, Reco_Insurance_Type is also categorized into two types namely Individual and Joint. Is_Spuse has two options i.e. Yes / No. While the health indicator is categorized into 9 options ranging from X1 - X9.

Data Visualization

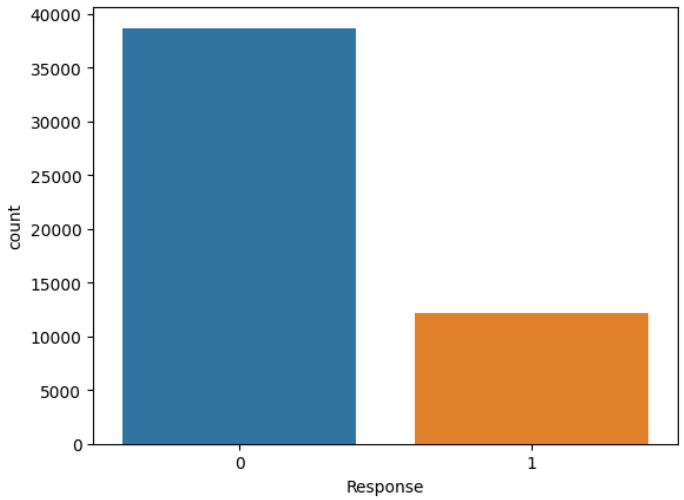
```
[9]  cols = ['Accomodation_Type', 'Reco_Insurance_Type', 'Is_Spouse', 'Health Indicator']
     fig, ax = plt.subplots(2,2, figsize=(12,10))

     sns.countplot(data=df, x=cols[0], ax = ax[0][0])
     sns.countplot(data=df, x=cols[1], ax = ax[0][1])
     sns.countplot(data=df, x=cols[2], ax = ax[1][0])
     sns.countplot(data=df, x=cols[3], ax = ax[1][1])
     plt.show()
```



This count plot shows the count of the target attribute i.e. the response attribute in the Health_Insurance dataset.

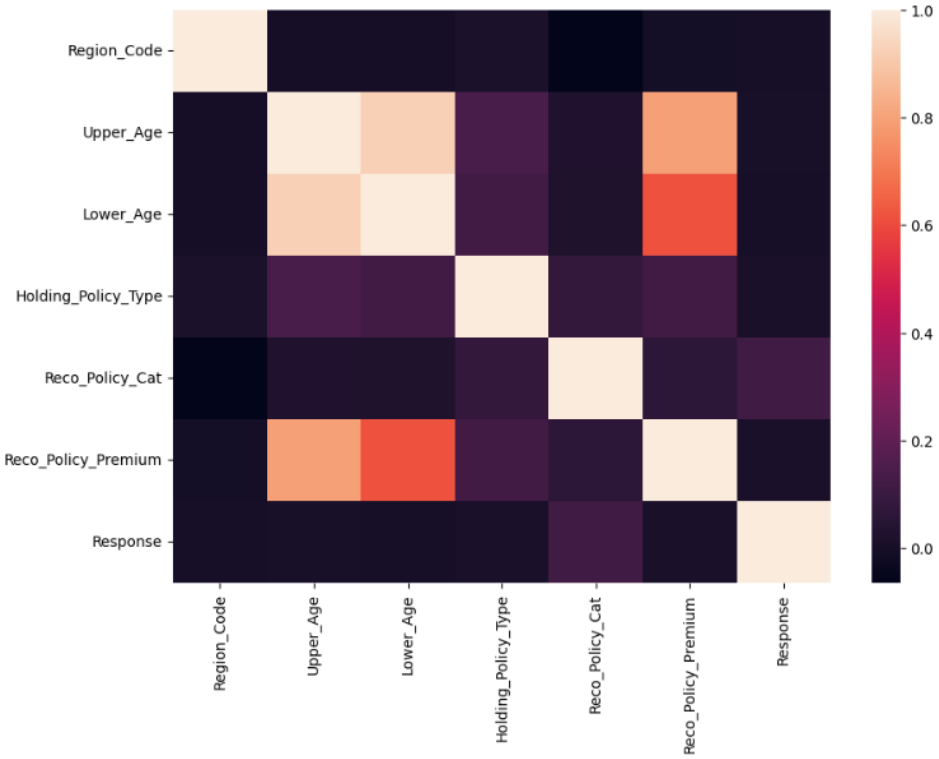
```
sns.countplot(x=df.Response)
plt.show()
```



This is the heat map showing the amount of correlation in each attribute.

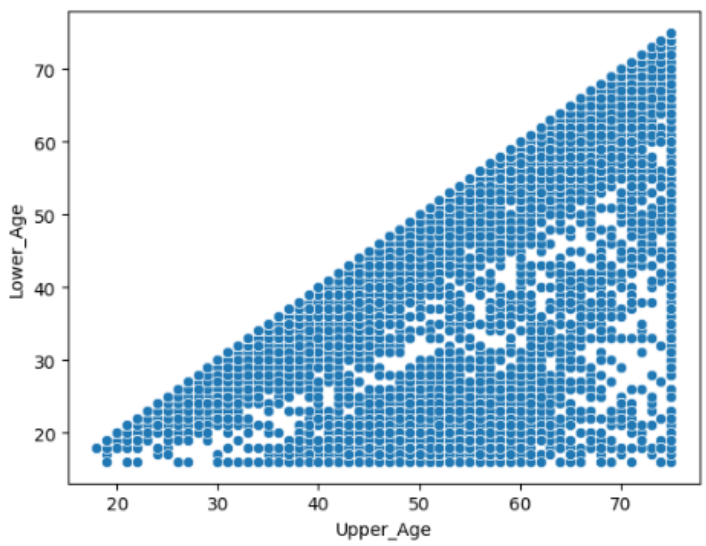
```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr())
plt.show()
```

```
<ipython-input-11-1f628e289267>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated.
  sns.heatmap(df.corr())
```



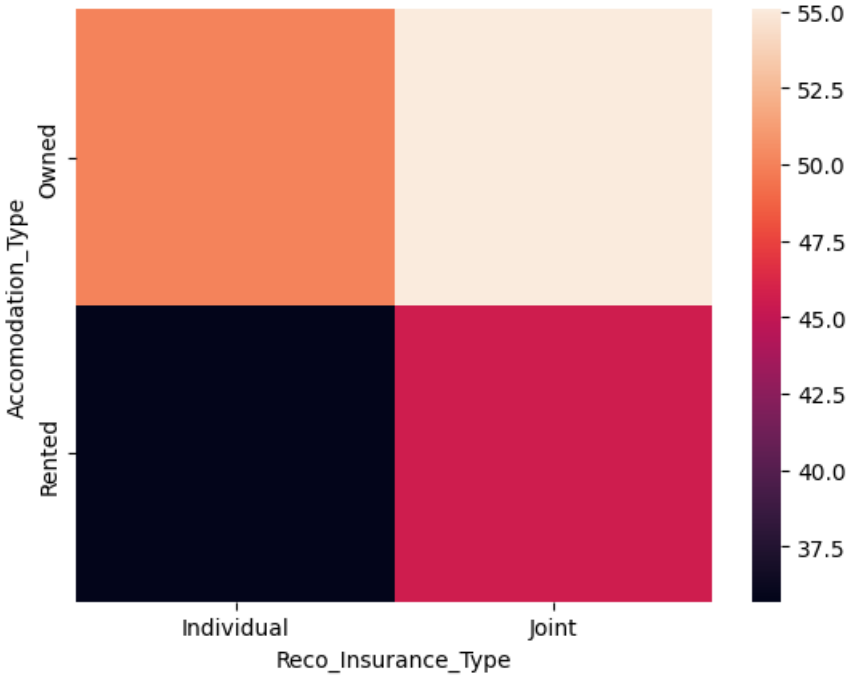
This is a scatterplot between the attributes namely the Upper_Age and Lower_Age.

```
sns.scatterplot(df, x= 'Upper_Age', y='Lower_Age')
plt.show()
```



This is a pivot table plotted between three attributes that are Accomdation_Type as index, Reco_Insurance_Type as columns and Upper_Age as values.

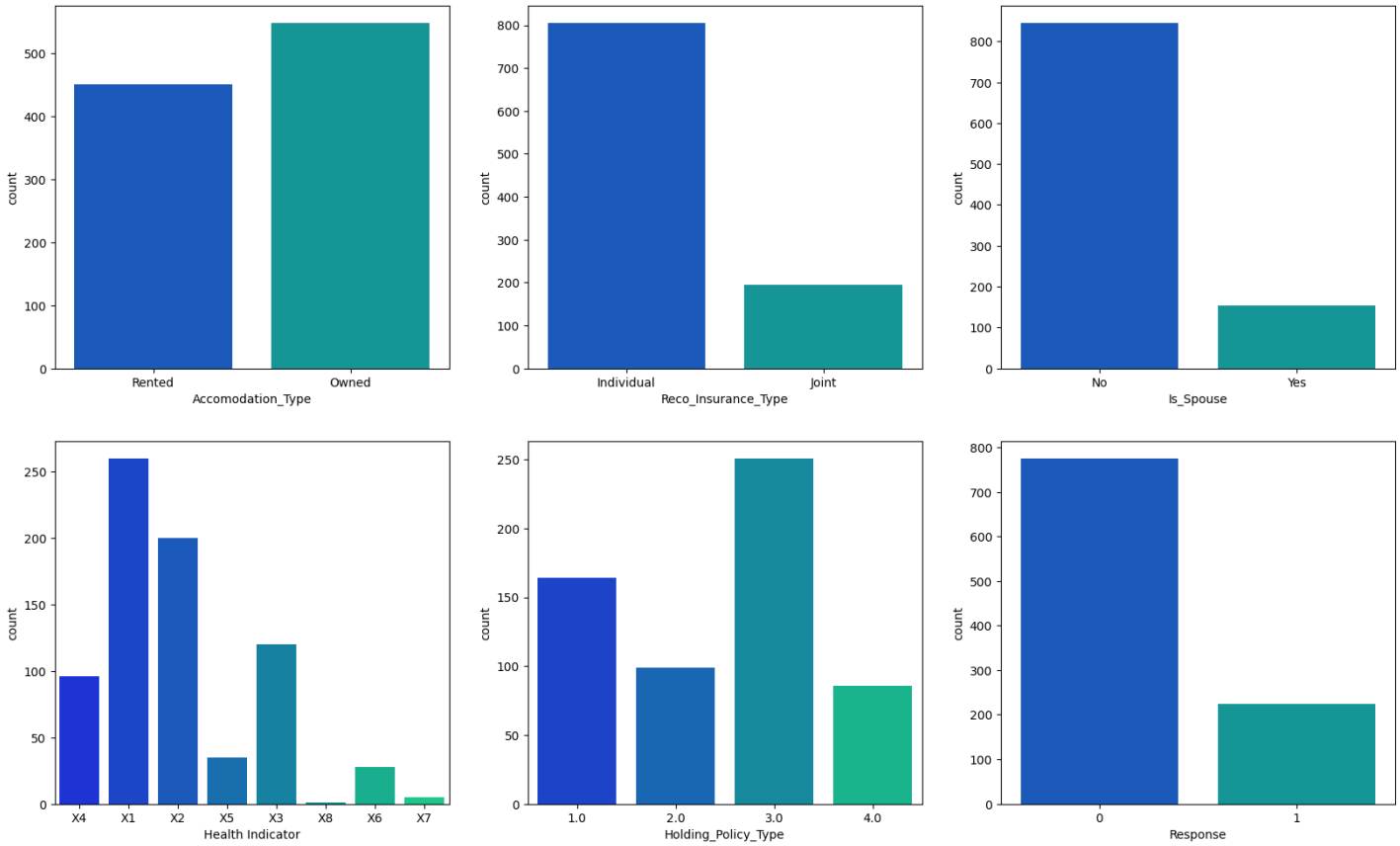
```
d1 = df.pivot_table(index='Accomodation_Type',columns= 'Reco_Insurance_Type', values='Upper_Age', aggfunc='mean')
sns.heatmap(data=d1)
plt.show()
```



These are the count plots made using the sample data from the dataset. The size of the sample data is 1000.

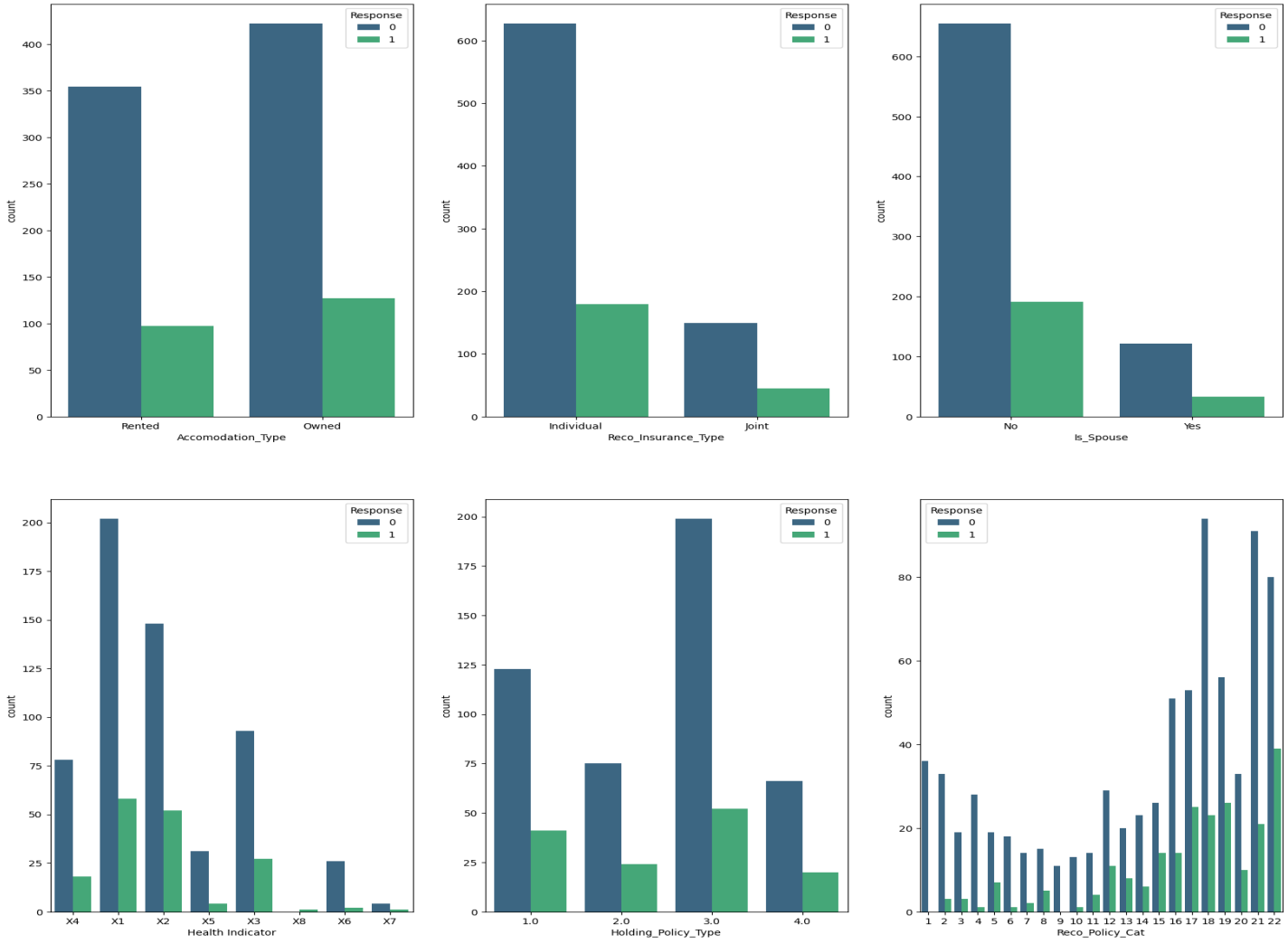
```
[15] sample_data = df.sample(1000)
```

```
fig, axes = plt.subplots(2, 3, figsize = (20, 12))
sns.countplot(x = 'Accomodation_Type', data = sample_data, palette= 'winter', ax = axes[0, 0]);
sns.countplot(x = 'Reco_Insurance_Type', data = sample_data, palette= 'winter', ax = axes[0, 1]);
sns.countplot(x = 'Is_Spouse', data = sample_data, palette= 'winter', ax = axes[0, 2]);
sns.countplot(x = 'Health Indicator', data = sample_data, palette= 'winter', ax = axes[1, 0]);
sns.countplot(x = 'Holding_Policy_Type', data = sample_data, palette= 'winter', ax = axes[1, 1]);
sns.countplot(x = 'Response', data = sample_data, palette= 'winter', ax = axes[1, 2]);
```



These are also the countplot but by using hue as the response attribute.

```
[17] fig, axes = plt.subplots(2, 3, figsize = (20, 20))
     sns.countplot(x = 'Accomodation_Type', data = sample_data, hue= 'Response', palette= 'viridis', ax = axes[0, 0]);
     sns.countplot(x = 'Reco_Insurance_Type', data = sample_data, hue= 'Response',  palette= 'viridis', ax = axes[0, 1]);
     sns.countplot(x = 'Is_Spouse', data = sample_data, hue= 'Response',  palette= 'viridis', ax = axes[0, 2]);
     sns.countplot(x = 'Health Indicator', data = sample_data, hue= 'Response',  palette= 'viridis', ax = axes[1, 0]);
     sns.countplot(x = 'Holding_Policy_Type', data = sample_data, hue= 'Response',  palette= 'viridis', ax = axes[1, 1]);
     sns.countplot(x = 'Reco_Policy_Cat', data = sample_data, hue= 'Response',  palette= 'viridis', ax = axes[1, 2]);
```



Data Mining

Model Training

```
[30] X = df.drop(['Response'], axis=1)
     X.head()
```

ID	Region_Code	Upper_Age	Lower_Age	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Accomodation_Type_Owned	Accomodation_Type_Rented	City_Code_C1	City_Code_C10
1	3213	36	36	3.0	22	-0.589278	0	1	0	0
4	4378	52	48	3.0	19	0.369526	1	0	0	0
5	2190	44	44	1.0	16	-0.780041	0	1	0	0
6	1785	52	52	1.0	22	-0.022599	0	1	0	0
8	3175	75	73	4.0	17	2.171803	1	0	1	0

```
[31] y = df['Response']
```

```
[32] from sklearn.model_selection import train_test_split
```

```
[33] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=3)
```

Logistic regression

Logistic Regression

```
[56] from sklearn.linear_model import LogisticRegression
```

```
[57] model = LogisticRegression(max_iter=1000, random_state=3)
```

```
[58] model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=1000, random_state=3)
```

```
[59] y_pred = model.predict(X_test)
```

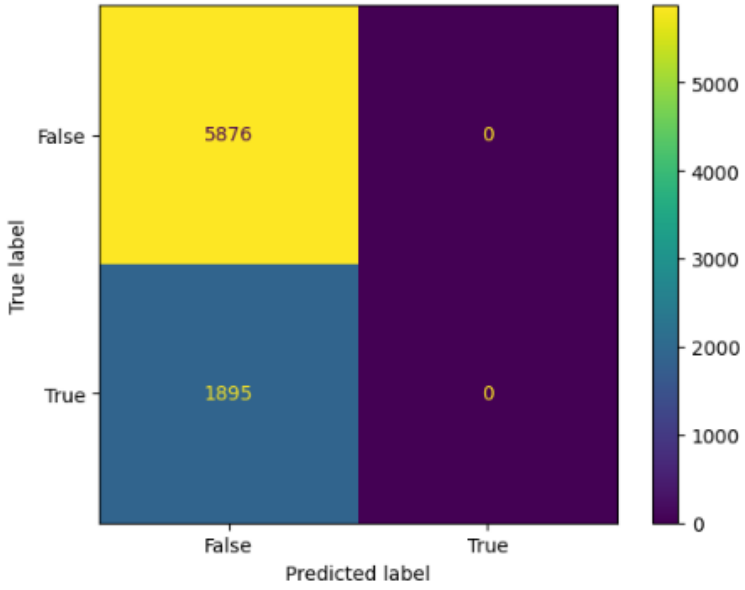
```
[60] logistic_score = model.score(X_test,y_test)
     logistic_score
```

```
0.75614464032943
```

```
[39] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
[40] confusion_matrix = confusion_matrix(y_test, y_pred)
```

```
[41] cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
     cm_display.plot()
     plt.show()
```



Decision Tree

Decision Tree

```
[42] from sklearn.tree import DecisionTreeClassifier
```

```
[43] model = DecisionTreeClassifier(criterion='entropy', random_state=3)
```

```
[44] model.fit(X_train, y_train)
```

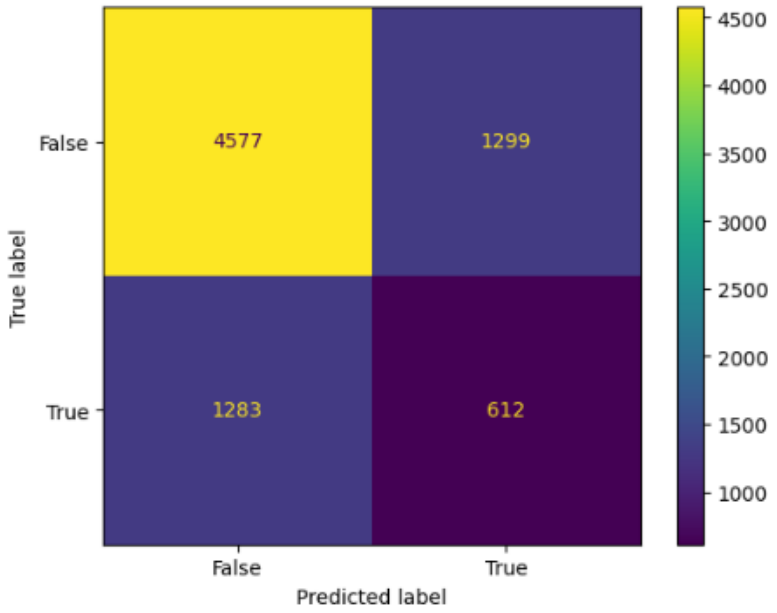
```
DecisionTreeClassifier(criterion='entropy', random_state=3)
```

```
[45] y_pred = model.predict(X_test)
```

```
[54] decision_score = model.score(X_test,y_test)
     decision_score
```

```
0.667739029725904
```

```
[47] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
     confusion_matrix = confusion_matrix(y_test, y_pred)
     cm_display = ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])
     cm_display.plot()
     plt.show()
```

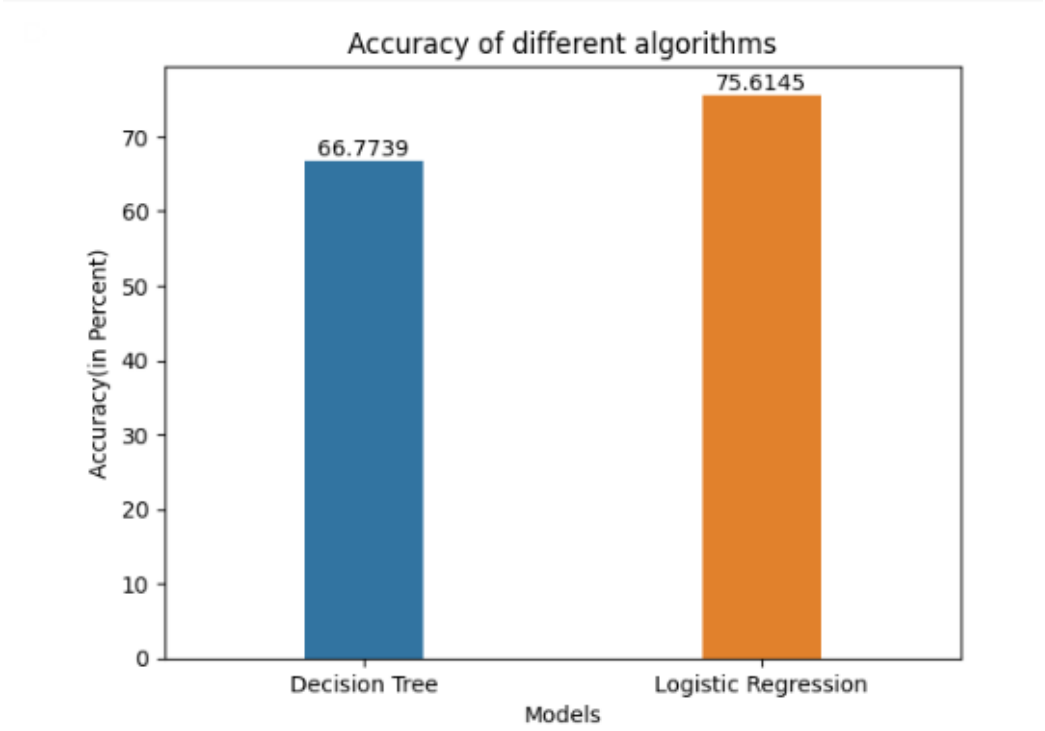


Results

For determining which customers will take the insurance policy, we can use different classification algorithms. For this problem we have used Decision Tree, and Logistic Regression.

According to the results, the accuracy of the Logistic Regression algorithm is more i.e. 75.6%. Therefore, for such type of dataset we will make use of Logistic Regression model to predict the outcome.

```
[61] ax = sns.barplot(x=['Decision Tree','Logistic Regression'], y=[decision_score*100, logistic_score*100], width=.3)
     plt.xlabel('Models')
     plt.ylabel('Accuracy(in Percent)')
     plt.title('Accuracy of different algorithms')
     for i in ax.containers:
         ax.bar_label(i)
     plt.show()
```



From the above comparison, we can see that Logistic Regression works best with the accuracy of almost 75.61% for the above dataset.

Link to Python Notebook - [Python Notebook]

Business Implications -

The business implications of our project are as follows -

Targeted marketing: To identify potential customers based on demographics such as age, location, and type of accommodation. This information can be used to target marketing efforts towards customers who are most likely to be interested in insurance products.

Customer retention: Insurance companies can use this project to identify customers who may be at risk of churning and take proactive measures to retain them.

Product development: Based on the type of insurance products that customers are interested in and results generated by the project. Insurance companies can use this information to develop new products that are tailored to customer needs and preferences.

Risk assessment: Health indicators and holding policy duration can be used to assess the risk of potential customers and adjust insurance premiums accordingly.

Operational efficiency: Based on holding policy type, Insurance companies can use this information to optimize their underwriting process and improve operational efficiency.

Conclusion -

In conclusion, the insurance lead prediction model is a useful business intelligence project for insurance companies to increase their customer base and revenue. By analyzing customer data and predicting the likelihood of a customer purchasing an insurance policy, companies can focus their marketing efforts on high-potential customers and improve their conversion rates.

However, to ensure the accuracy and reliability of the model, it is important to continuously monitor and update it with new data. Additionally, it is important to maintain data privacy and security, as the model relies on sensitive customer information.