Exploratory Data Analysis of Potential Insurance Buyers

The Data is about the potential insurance customers. It has the following features

- 'ID' : a serial number assigned to a customer
- 'City_Code' : code of the city individual/joint account holders is living.
- 'Region_Code' : code of region
- 'Accomodation_Type' : the type of their residence. (Rented/owned)
- 'Reco_Insurance_Type' : the type of insurance they had. (joint / individual)
- 'Upper_Age' : Max age amoung the joint account holders. (if individual, upper_age = lower_age)
- 'Lower_Age' : Min age among the joint account holders.
- 'Is_Spouse' : For joint policy, whether the other person is spouse or not.
- 'Health Indicator' : Some domain specific health indication
- 'Holding_Policy_Duration': the duration of policy the customer is using
- 'Holding_Policy_Type' : the type of policy selected
- 'Reco_Policy_Cat' : Recovered policy category, domain specific.
- 'Reco_Policy_Premium' : Recovered policy premium.
- 'Response' : Response given by customer at the end. 1 = accepting to take new policy, 0 = not willing to take.

Initial plan for Data Exploration:

1. Removing the unnecessary features first.
2. Replacing the null values and Data cleaning.
3. Data wrangling: mean, std, max and, also correlation
4. Univariate analysis: to understand single feature precisely.
5. Bi-variate and Multi-variate analysis: for drawing out insights, inferences.

Sample data:

```
df1
```

City_Code	Accomodation_Type	Reco_Insurance_Type	Upper_Age	Lower_Age	Is_Spouse	Health Indicator	Holding_Policy_Duration	Holding_Policy_Type	Reco_Policy_
C3	Rented	Individual	36	36	No	X1	15	3.0	
C5	Owned	Joint	75	22	No	X2	0	0.0	
C5	Owned	Individual	32	32	No	X10	1.0	1.0	
C24	Owned	Joint	52	48	No	X1	15	3.0	
C8	Rented	Individual	44	44	No	X2	3.0	1.0	
...	...	...	...	...	...	...	...	...	
C4	Rented	Individual	22	22	No	X3	0	0.0	
C5	Rented	Individual	27	27	No	X3	7.0	3.0	
C1	Rented	Individual	63	63	No	X2	15	1.0	
C1	Owned	Joint	71	49	No	X2	2.0	2.0	
C3	Rented	Individual	24	24	No	X3	2.0	3.0	

ows × 12 columns

Data Cleaning:

Removing ID for now and also Region codes since it has way higher unique variables which is not helpful for further analysis.

```
df1 = df.drop(['ID', 'Region_Code'], axis = 1)
```

Replacing the null values:

First, let's see how many null values do we have in the df1.

```
df1.isnull().sum()
```

```
City_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
```

So, we have nulls in only three columns. Will do following operations.

```
df1 = df1.replace({'Holding_Policy_Duration' : { np.nan: 0, '14+' : 15}})
df1['Holding_Policy_Type'] = df1['Holding_Policy_Type'].replace(np.nan, 0)
df1['Health Indicator'] = df1['Health Indicator'].replace(np.nan, "X10")
```

Let's see the datatypes of each column.

```
df1.dtypes
```

```
City_Code                  object
Accomodation_Type          object
Reco_Insurance_Type        object
Upper_Age                   int64
Lower_Age                   int64
Is_Spouse                  object
Health Indicator           object
Holding_Policy_Duration    object
Holding_Policy_Type       float64
Reco_Policy_Cat             int64
Reco_Policy_Premium       float64
Response                    int64
dtype: object
```

Everything looks fine except Holding_policy_duration which is numerical. So it should be int.

```
df1['Holding_Policy_Duration']= pd.to_numeric(df1['Holding_Policy_Duration'])
```

Let’s understand the correlation among different features with help of spearman coefficient.

```
df1.corr(method='spearman')
```

	Upper_Age	Lower_Age	Holding_Policy_Type	Reco_Policy_Cat	Reco_Policy_Premium	Response
Upper_Age	1.000000	0.909629	0.318290	0.024441	0.835294	0.004385
Lower_Age	0.909629	1.000000	0.287866	0.021338	0.669743	-0.000472
Holding_Policy_Type	0.318290	0.287866	1.000000	0.050934	0.265066	0.005052
Reco_Policy_Cat	0.024441	0.021338	0.050934	1.000000	0.061397	0.092954
Reco_Policy_Premium	0.835294	0.669743	0.265066	0.061397	1.000000	0.007918
Response	0.004385	-0.000472	0.005052	0.092954	0.007918	1.000000

- Response is highly correlated to Reco_Policy_Cat.
- Upper age is highly correlated to Reco_policy_premium.
- Both ages are highly correlated. obviously.

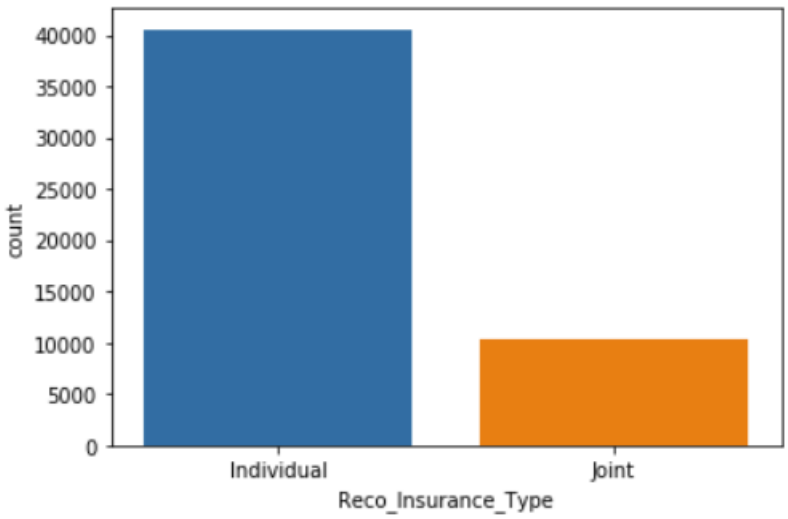
In further study, let’s also plot the above correlation.

Univariate Analysis:

Hypothesis test-1:

Null hypothesis: Individual policy takers are more than the Joint policy holders.

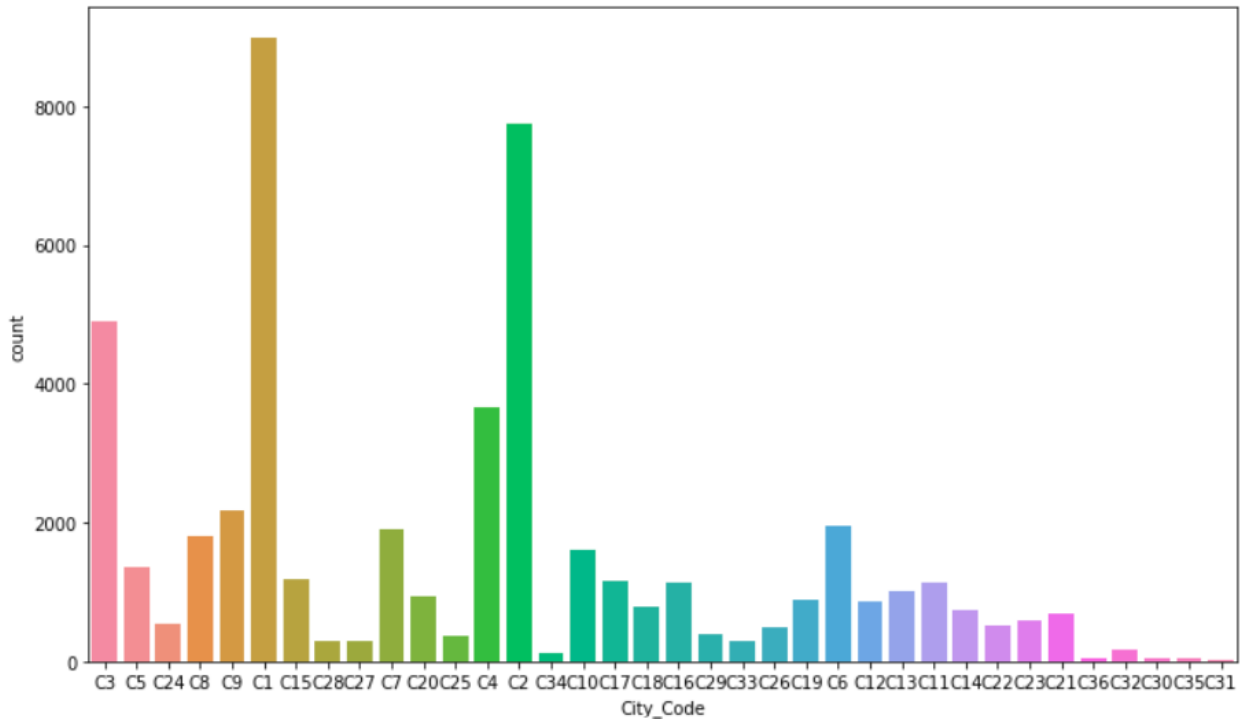
For evidence, let’s plot the “Reco_Insurance_Type”



Hence, Null-hypothesis is true. Individual policy takers are way more than the joint policy holders.

Also, let’s see the univariate plots of other features.

```
countplt, ax = plt.subplots(figsize = (12,7))
ax =sns.countplot(x = 'City_Code', data=df1)
```



The City C1, C2 are the highest among the policy holders.

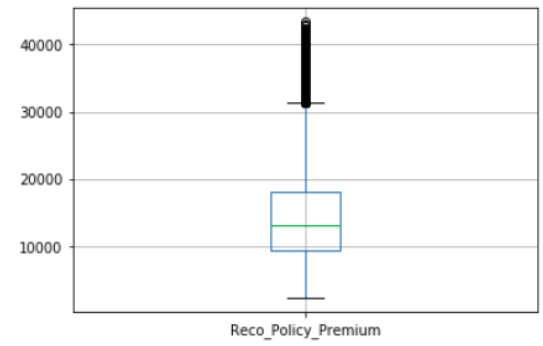
Let’s understand the recovered policy premium now.

```
df1['Reco_Policy_Premium'].describe()
```

```
count    50882.000000
mean     14183.950069
std       6590.074873
min       2280.000000
25%       9248.000000
50%      13178.000000
75%      18096.000000
max      43350.400000
```

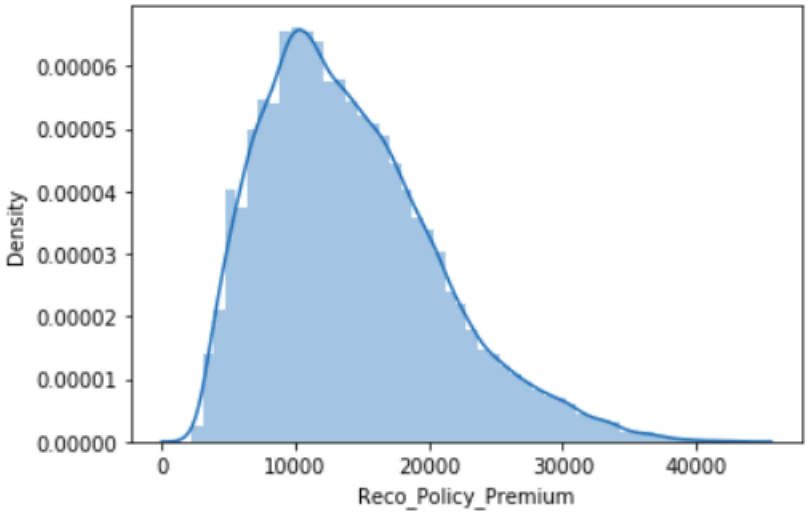
```
df1.boxplot(column=['Reco_Policy_Premium'])
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x2cee76ed188>
```



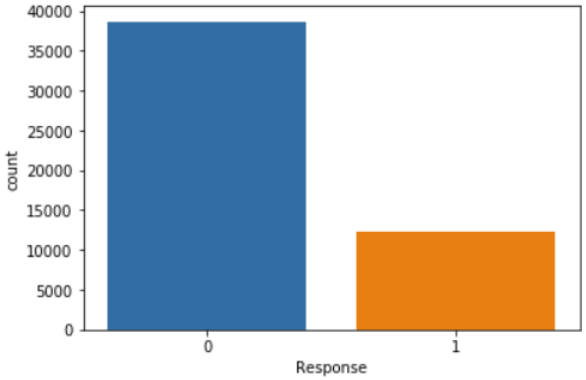
```
sns.distplot(df1['Reco_Policy_Premium'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2d413999548>
```



The curve is skewed towards right with a mean of 14183.95 of recovered policy premium. So very few recovered premium even above 30000.

Now, let's see about the target variable i.e., Response.



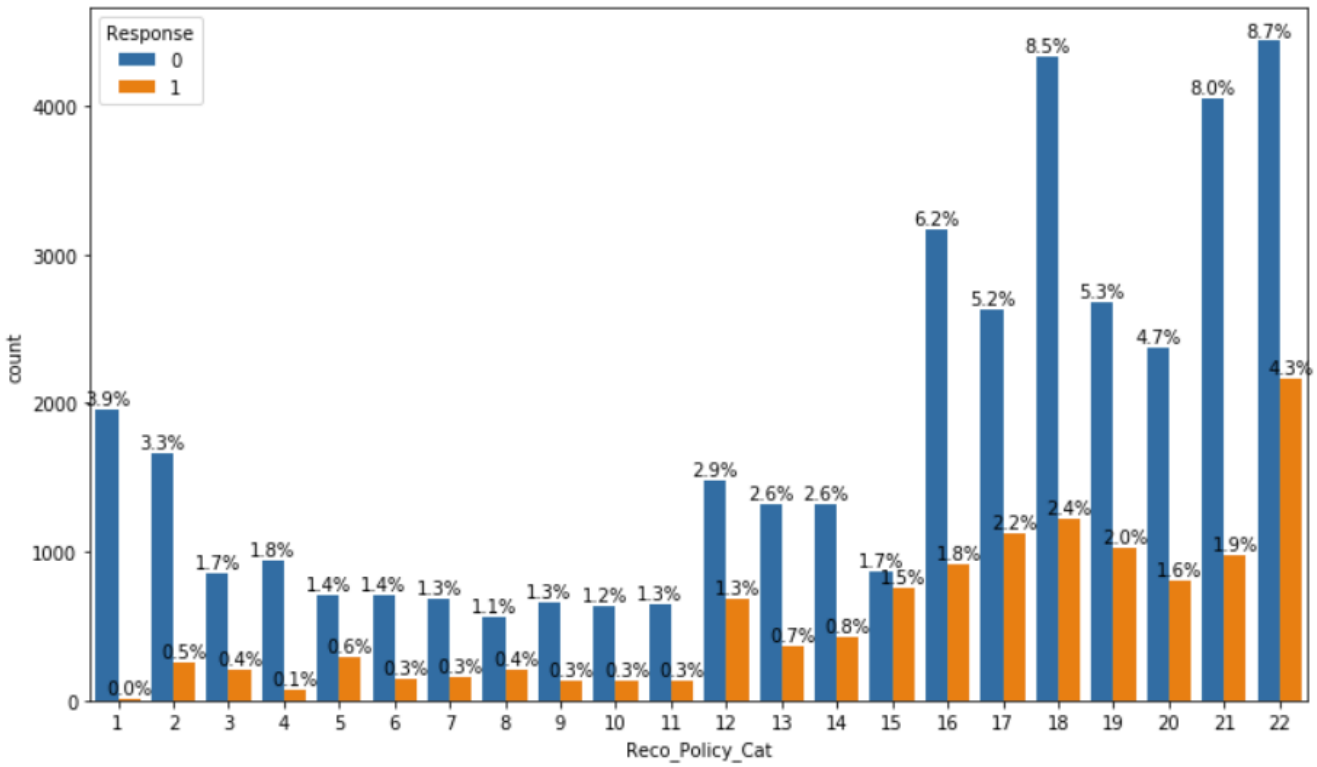
It is clearly understood that the data is biased towards 0.

Hypothesis testing -2:

Null-hypothesis: above 50% of policy holders are from policies 15 to 22.

```
fig, ax = plt.subplots(figsize = (12,7))

ax= sns.countplot('Reco_Policy_Cat', data = df1, hue= 'Response')
total = len(df1)
for p in ax.patches:
    percentage = f'{100 * p.get_height() / total:.1f}%\n'
    x = p.get_x() + p.get_width() / 2
    y = p.get_height()
    ax.annotate(percentage, (x, y), ha='center', va='center')
```



From the figure, we could calculate that around 62% of total policy holders are from category 15 to 22.

Hence Null hypothesis is true.

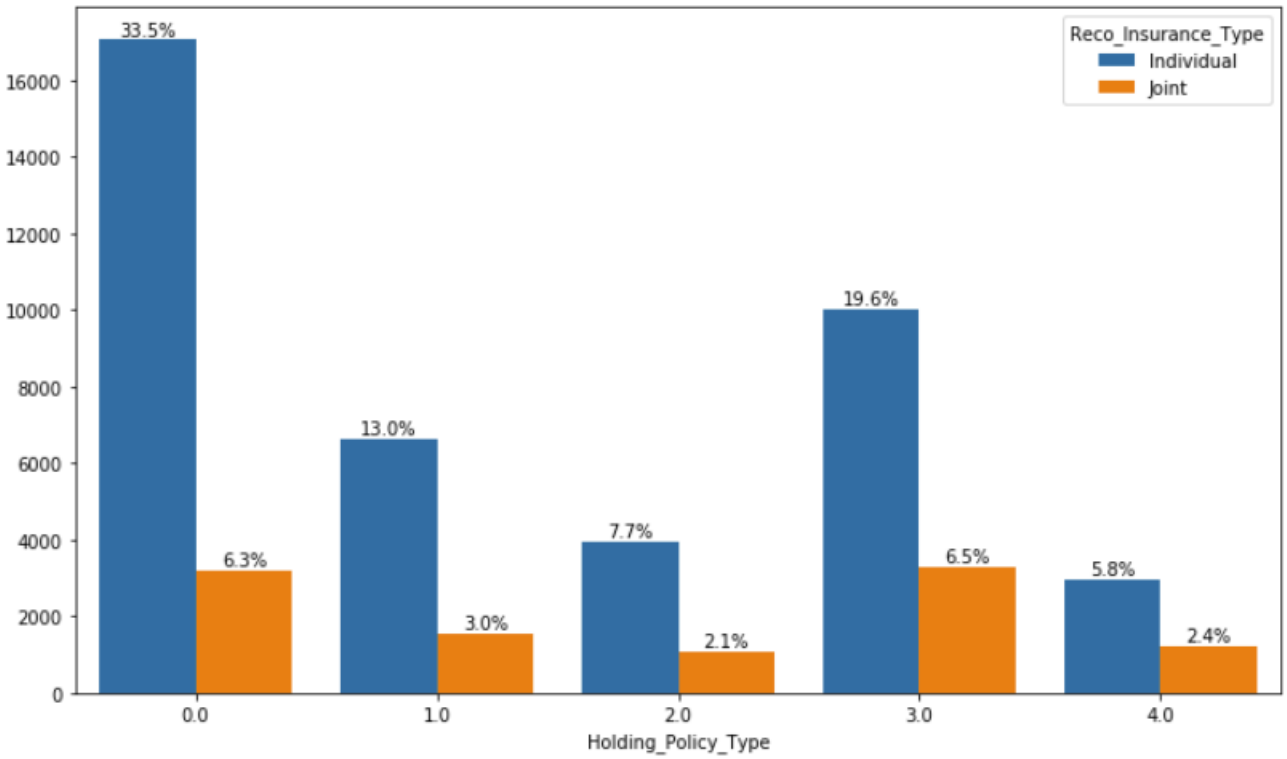
Hypothesis Testing – 3:

Null hypothesis: The highest policy taken is Policy 0.

```
df1.isnull().sum()
```

```
City_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
```

the null values are replaced with ‘0’ in data cleaning.



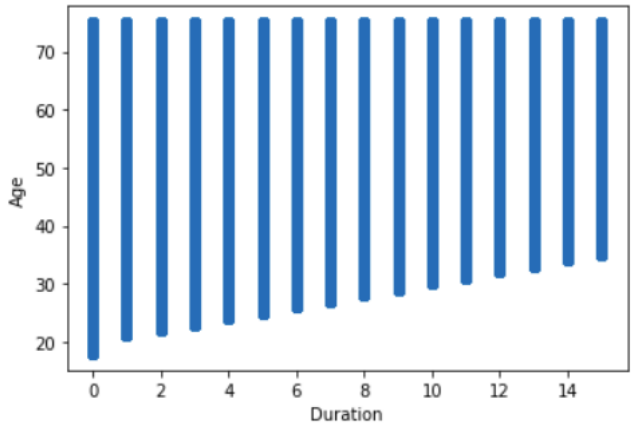
We see that the policy 0 is high. But we know that It's just the replaced values.

Since, we do not have evidence to prove that other policies are higher.

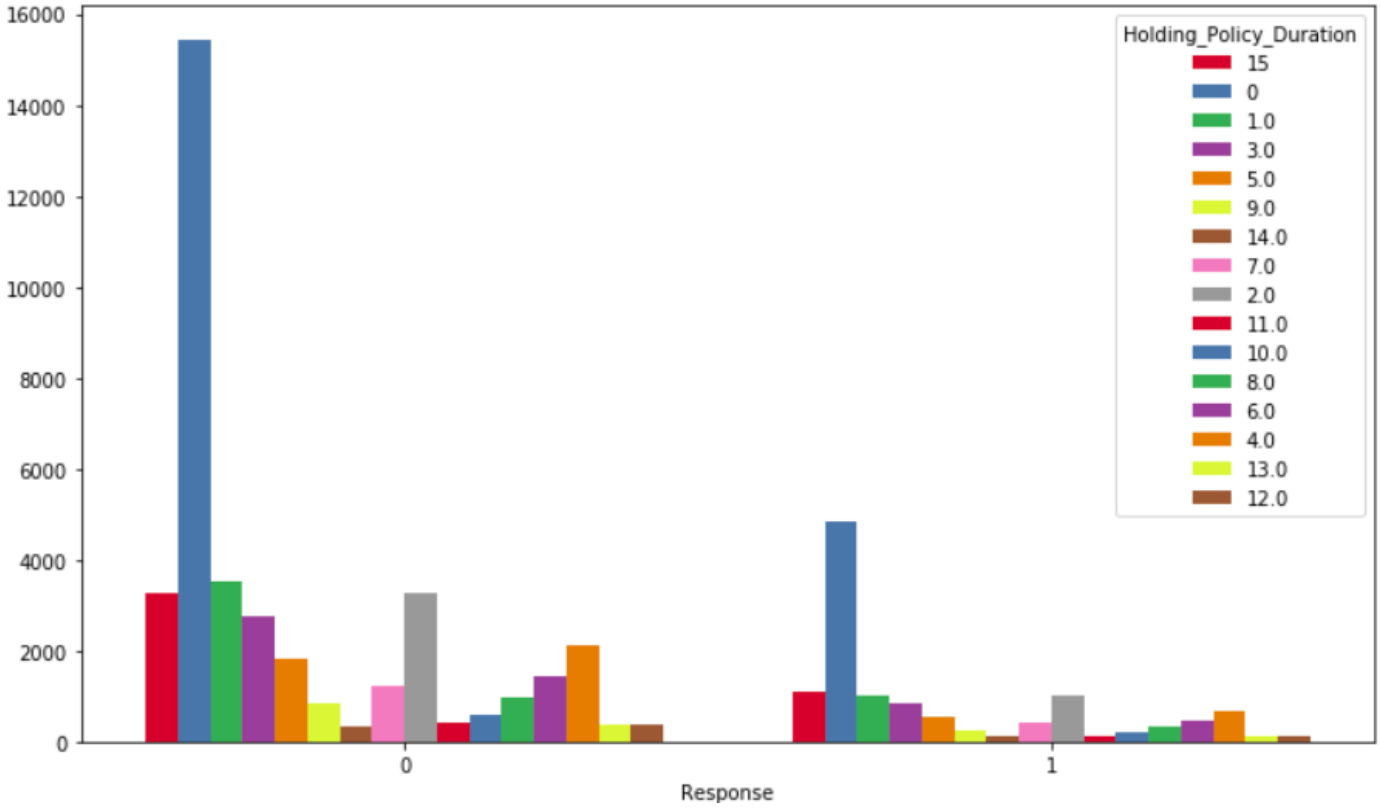
This is a Type-2 error.

Multi-Variate Analysis:

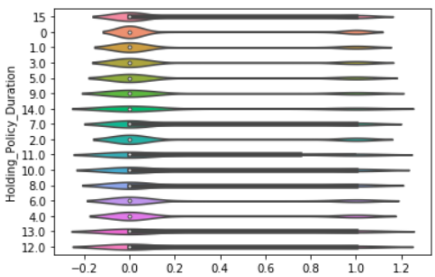
Let’s analyze the policy duration and upper age with the help of scatter plot.



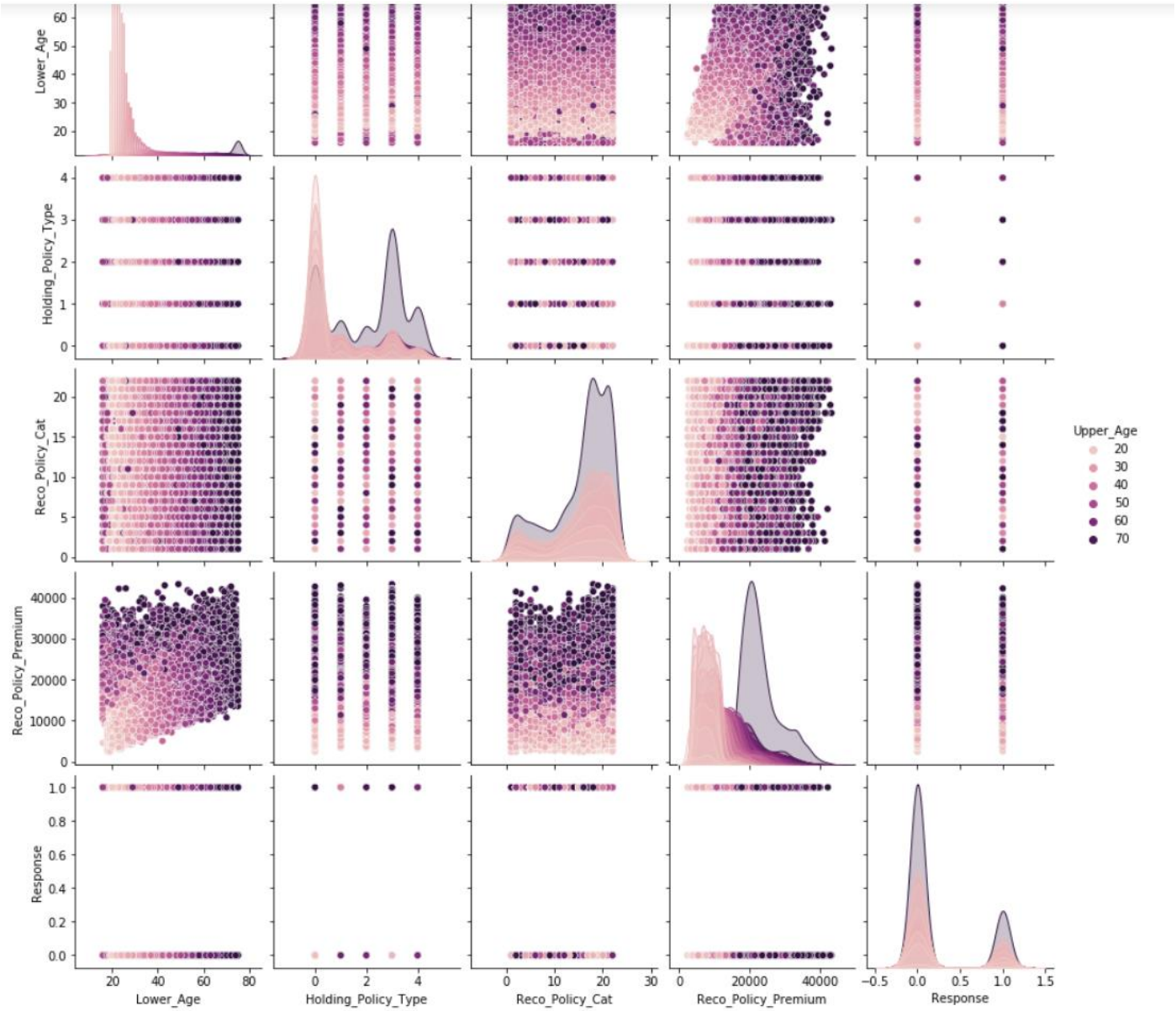
Policy duration vs response:



The policy duration is zero for most. Next to it, policy holders with a policy_duration of 2 years is the highest. We will see the same in violin plot.



Now, we try to understand whole data with respect to upper age using pairplots.



Insights from above graph, (older people=above age 50)

- The older people are more willing to take policy 3 and policy 4.
- older people are the one who recovered more premium with a mean of 20000.

Further Study:

- The data should is highly biased, so a oversampling or undersampling to be done. SMOTE method would be good.
- If not, we could XgbClassifier or Randomforest classifier which are less effected by imbalanced data.
- Labelencoding/ Onehotencoding to be done for the categorical features.
- Hyperparmeter tuning using gridsearch will also result in high performance.