In [1]:
```
## Import libraries and packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
pd.set_option("display.max_columns",None)
pd.set_option("display.max_rows",None)
from scipy.stats import kruskal
from scipy.stats import mannwhitneyu
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from math import sqrt
import statsmodels.api as sm
```

In [2]:
```
## load the dataset
data=pd.read_csv("AutoInsurance.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	Customer	State	Customer Lifetime Value	Response	Coverage	Education	Effective To Date	Empl
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	3/2/2011	

In [4]:
```
data.info()
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9134 entries, 0 to 9133
Data columns (total 24 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Customer                       9134 non-null   object 
 1   State                          9134 non-null   object 
 2   Customer Lifetime Value        9134 non-null   float64
 3   Response                       9134 non-null   object 
 4   Coverage                       9134 non-null   object 
 5   Education                      9134 non-null   object 
 6   Effective To Date              9134 non-null   object 
 7   EmploymentStatus               9134 non-null   object 
 8   Gender                         9134 non-null   object 
 9   Income                         9134 non-null   int64  
 10  Location Code                  9134 non-null   object 
 11  Marital Status                 9134 non-null   object 
 12  Monthly Premium Auto           9134 non-null   int64  
 13  Months Since Last Claim        9134 non-null   int64  
 14  Months Since Policy Inception  9134 non-null   int64  
 15  Number of Open Complaints      9134 non-null   int64  
 16  Number of Policies             9134 non-null   int64  
 17  Policy Type                    9134 non-null   object 
 18  Policy                         9134 non-null   object 
 19  Renew Offer Type               9134 non-null   object 
 20  Sales Channel                  9134 non-null   object 
 21  Total Claim Amount             9134 non-null   float64
 22  Vehicle Class                  9134 non-null   object 
 23  Vehicle Size                   9134 non-null   object 
dtypes: float64(2), int64(6), object(16)
memory usage: 1.7+ MB
```

In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Customer                         0
State                            0
Customer Lifetime Value          0
Response                         0
Coverage                         0
Education                        0
Effective To Date                0
EmploymentStatus                 0
Gender                           0
Income                           0
Location Code                    0
Marital Status                   0
Monthly Premium Auto             0
Months Since Last Claim          0
Months Since Policy Inception    0
Number of Open Complaints        0
Number of Policies               0
Policy Type                      0
Policy                           0
Renew Offer Type                 0
Sales Channel                    0
Total Claim Amount               0
Vehicle Class                    0
Vehicle Size                     0
dtype: int64
```

In [6]:
```
data.describe(include="all")
```

Out[6]:

	Customer	State	Customer Lifetime Value	Response	Coverage	Education	Effective To Date	E
count	9134	9134	9134.000000	9134	9134	9134	9134	
unique	9134	5	NaN	2	3	5	59	
top	BU79786	California	NaN	No	Basic	Bachelor	10/1/2011	
freq	1	3150	NaN	7826	5568	2748	195	
mean	NaN	NaN	8004.940475	NaN	NaN	NaN	NaN	
std	NaN	NaN	6870.967608	NaN	NaN	NaN	NaN	
min	NaN	NaN	1898.007675	NaN	NaN	NaN	NaN	
25%	NaN	NaN	3994.251794	NaN	NaN	NaN	NaN	
50%	NaN	NaN	5780.182197	NaN	NaN	NaN	NaN	
75%	NaN	NaN	8962.167041	NaN	NaN	NaN	NaN	
max	NaN	NaN	83325.381190	NaN	NaN	NaN	NaN	

In [7]:
```
data.describe()
```

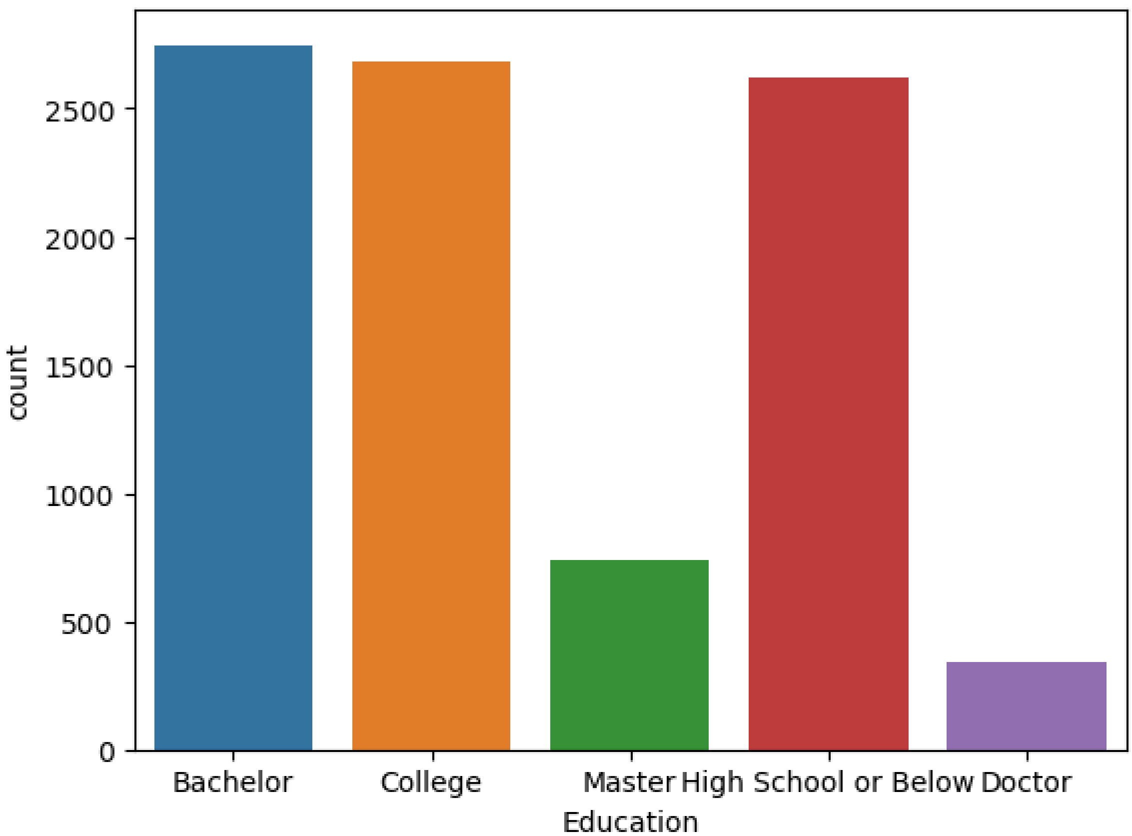
Loading [MathJax]/extensions/Safe.js

Out[7]:

	Customer Lifetime Value	Income	Monthly Premium Auto	Months Since Last Claim	Months Since Policy Inception	Number of Open Complaints
count	9134.000000	9134.000000	9134.000000	9134.000000	9134.000000	9134.000000
mean	8004.940475	37657.380009	93.219291	15.097000	48.064594	0.384388
std	6870.967608	30379.904734	34.407967	10.073257	27.905991	0.910384
min	1898.007675	0.000000	61.000000	0.000000	0.000000	0.000000
25%	3994.251794	0.000000	68.000000	6.000000	24.000000	0.000000
50%	5780.182197	33889.500000	83.000000	14.000000	48.000000	0.000000
75%	8962.167041	62320.000000	109.000000	23.000000	71.000000	0.000000
max	83325.381190	99981.000000	298.000000	35.000000	99.000000	5.000000

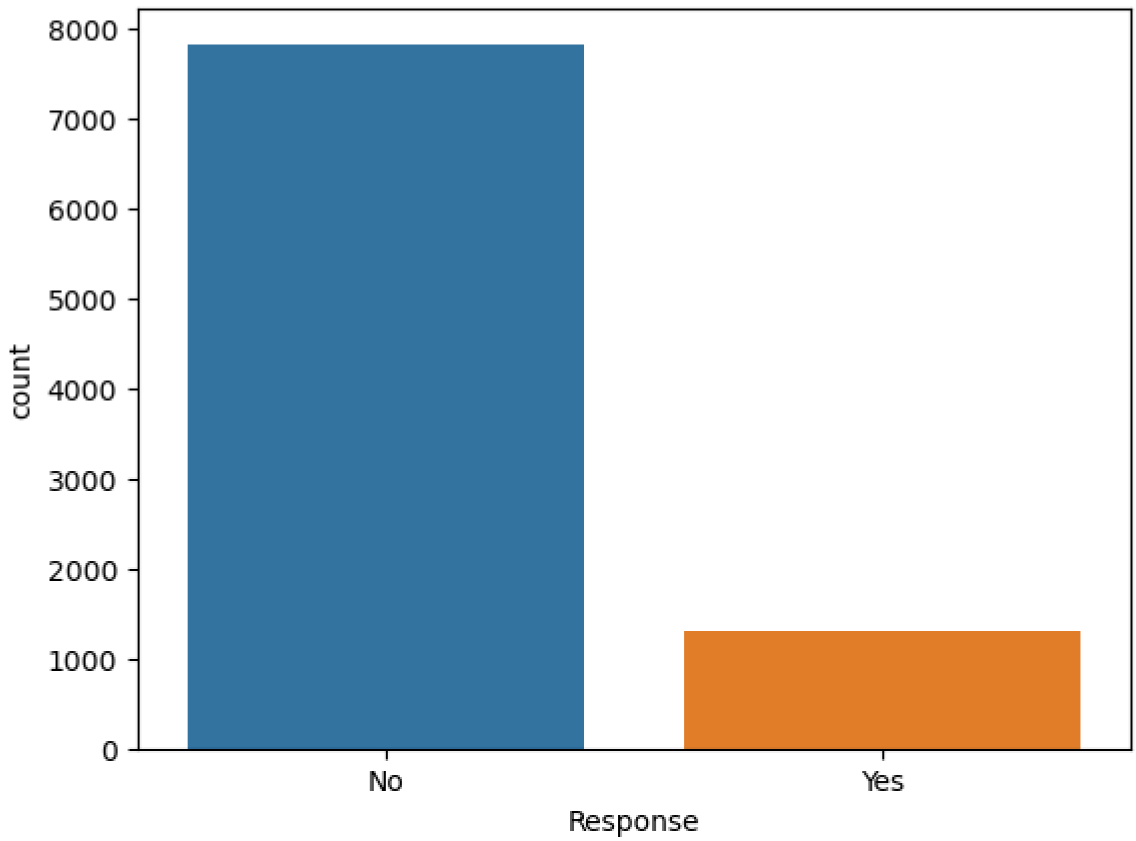
In [10]:

```
sns.countplot(x="Education",data=data)
plt.show()
```

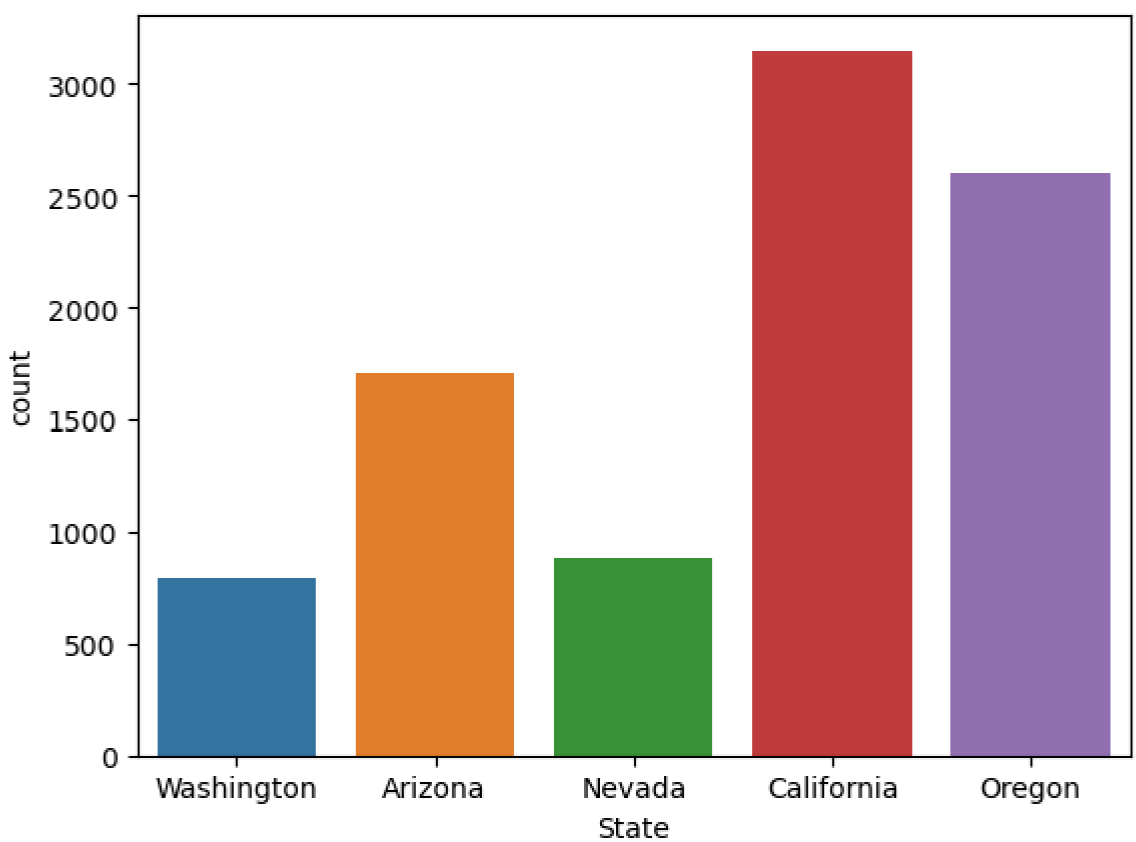


In [8]:

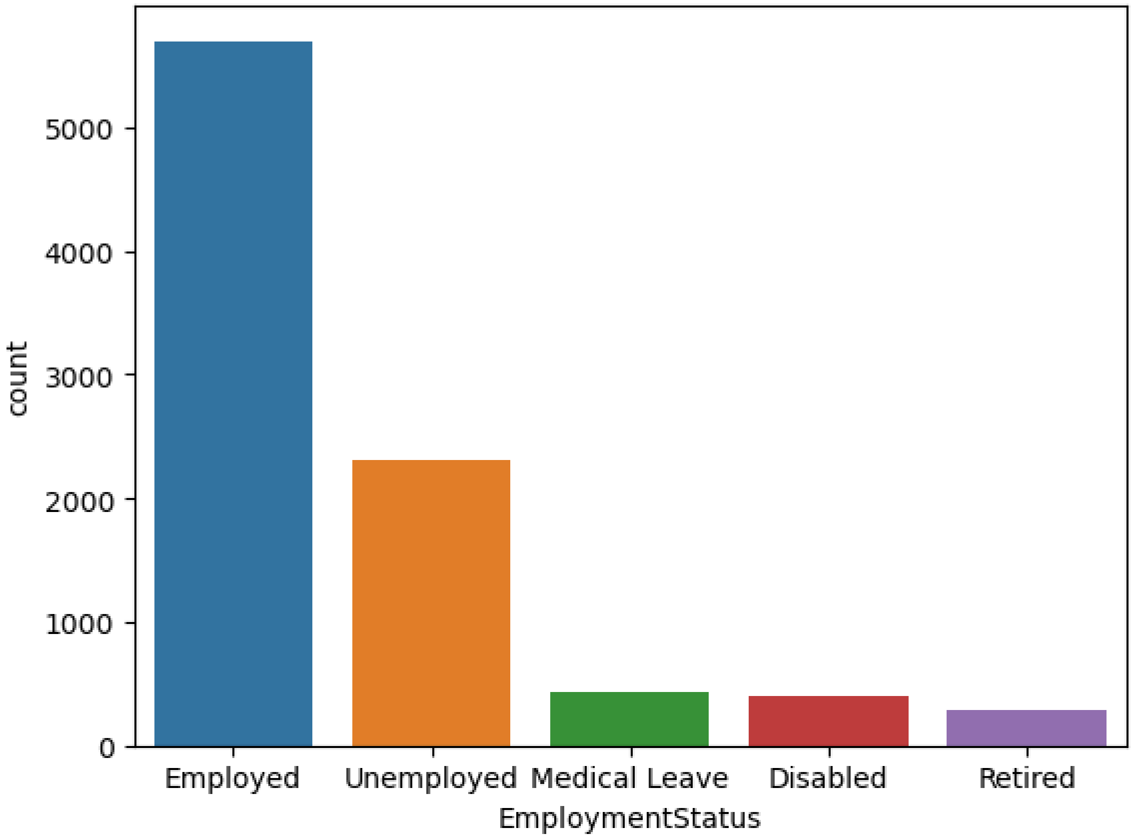
```
sns.countplot(x="Response",data=data)
plt.show()
```



In [11]:
```
sns.countplot(x="State",data=data)
plt.show()
```

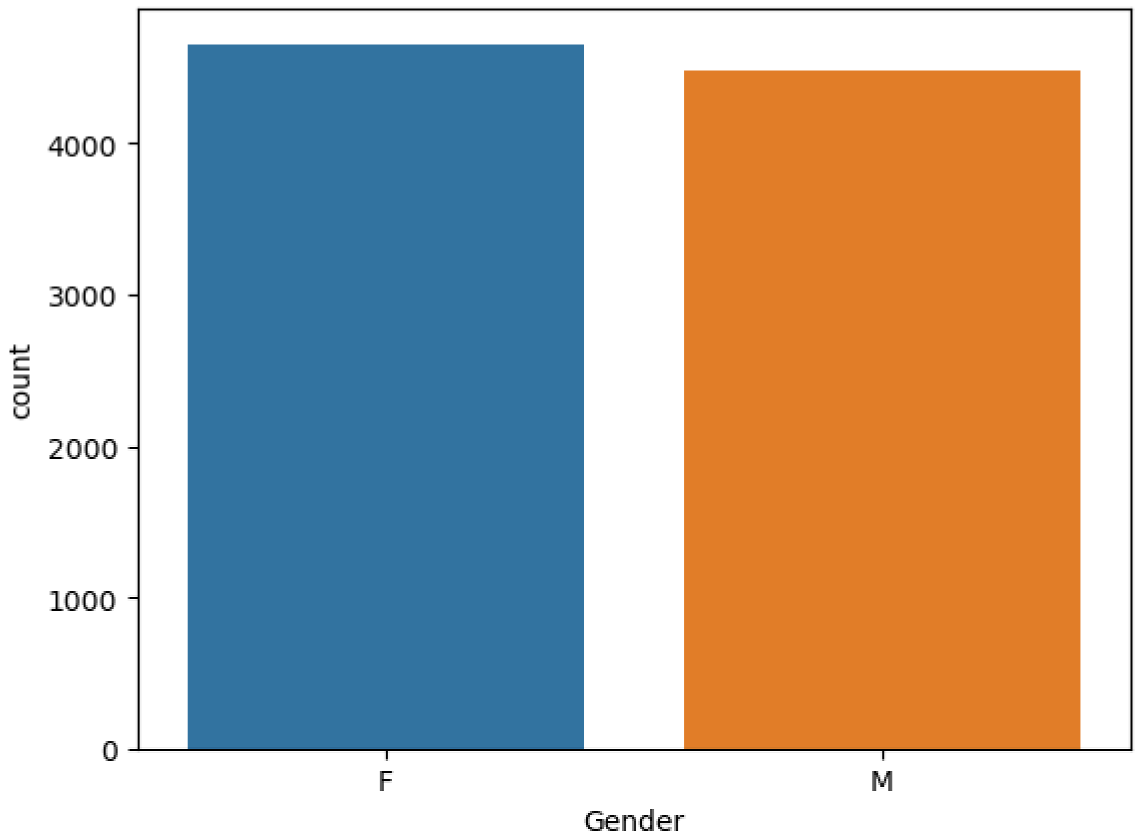


In [12]:
```python
sns.countplot(x="EmploymentStatus",data=data)
plt.show()
```

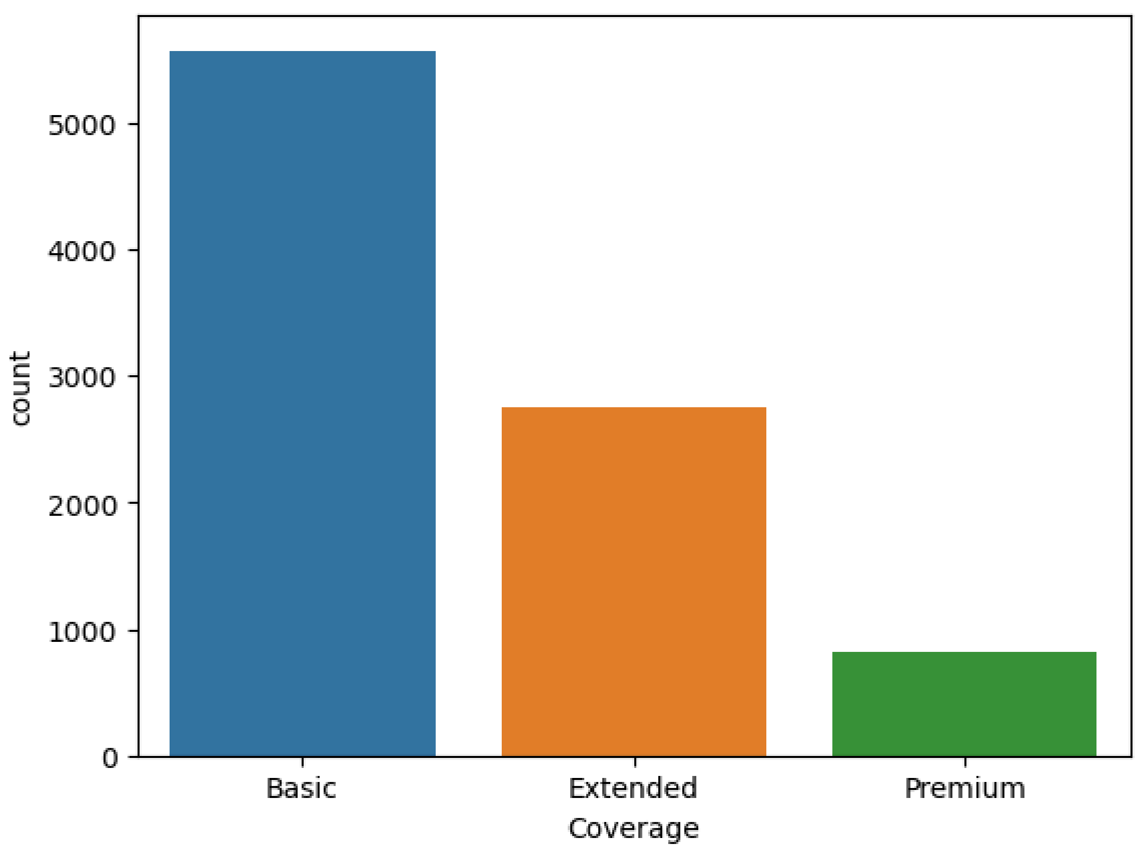


In [13]:
```python
sns.countplot(x="Gender",data=data)
plt.show()
```

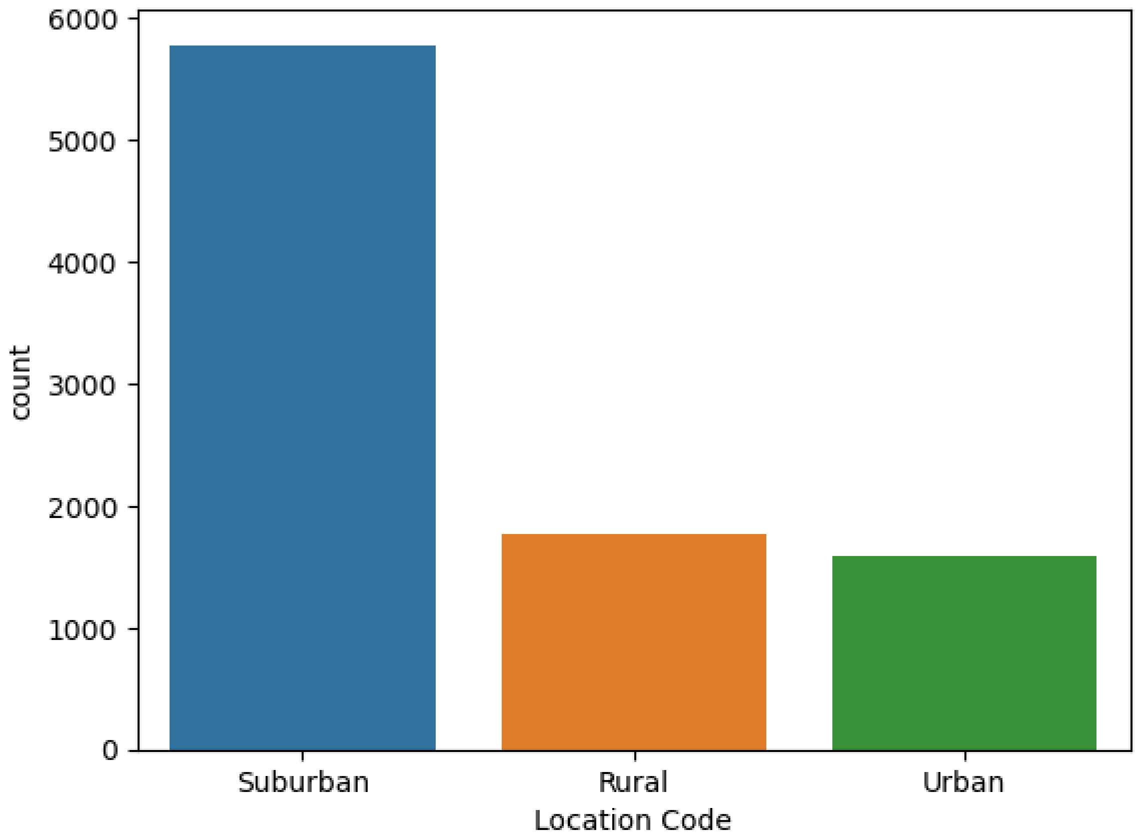
Loading [MathJax]/extensions/Safe.js



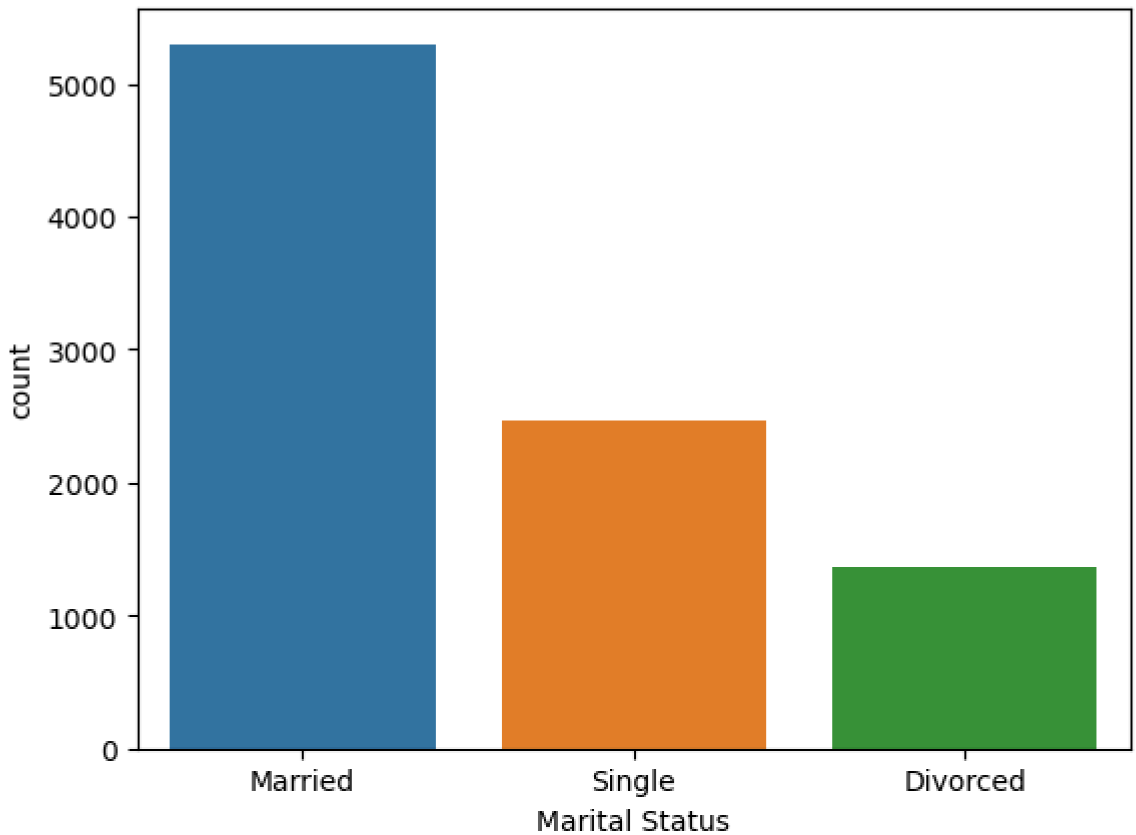
In [9]:
```
sns.countplot(x="Coverage",data=data)
plt.show()
```



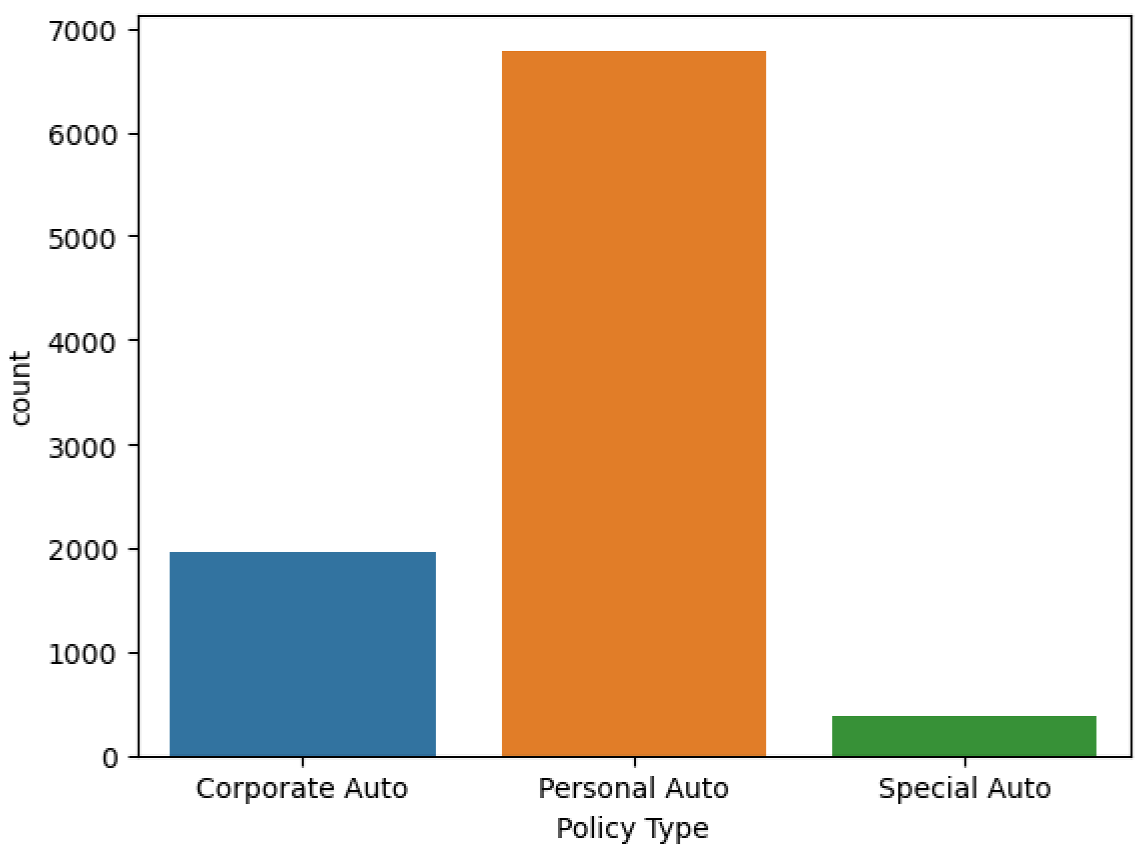
In [14]:
```
sns.countplot(x="Location Code",data=data)
plt.show()
```



In [15]:
```
sns.countplot(x="Marital Status",data=data)
plt.show()
```

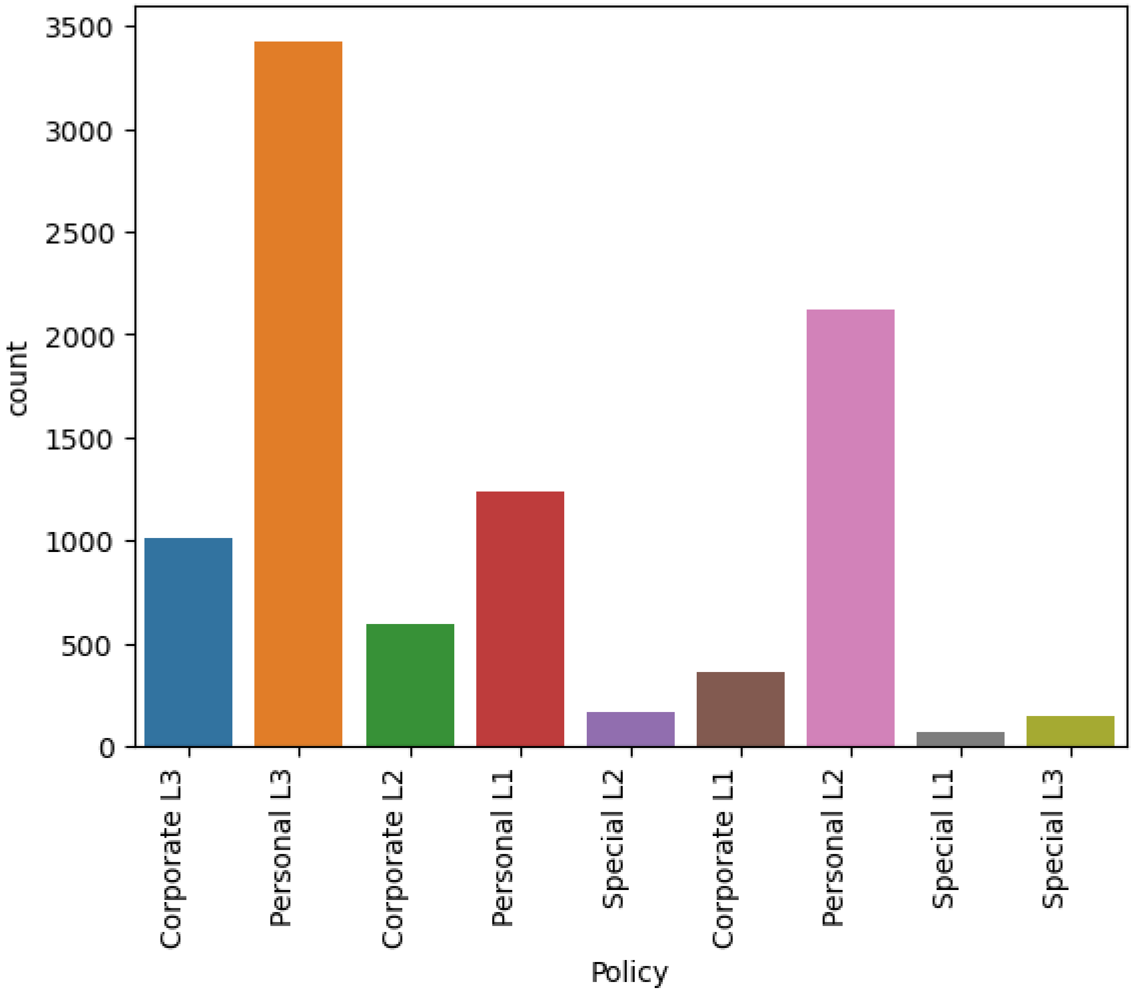


In [16]:
```
sns.countplot(x="Policy Type",data=data)
plt.show()
```

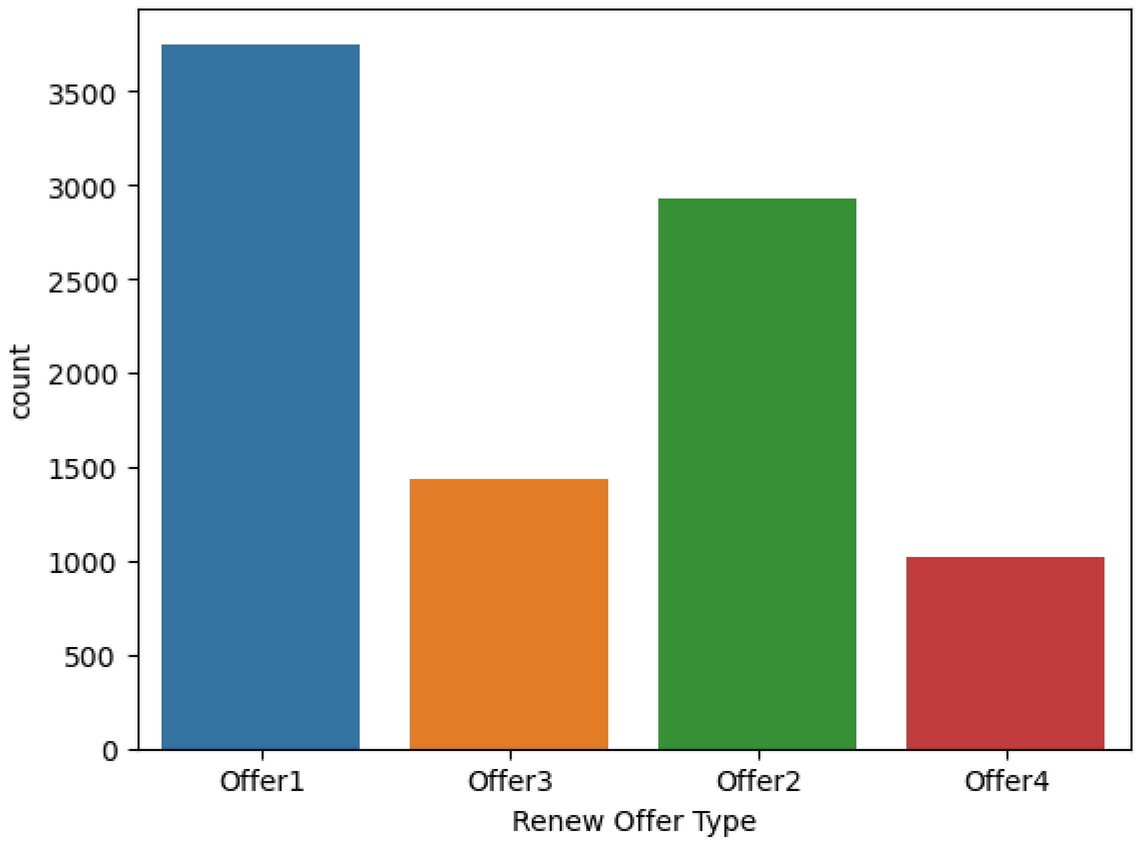


Loading [MathJax]/extensions/Safe.js

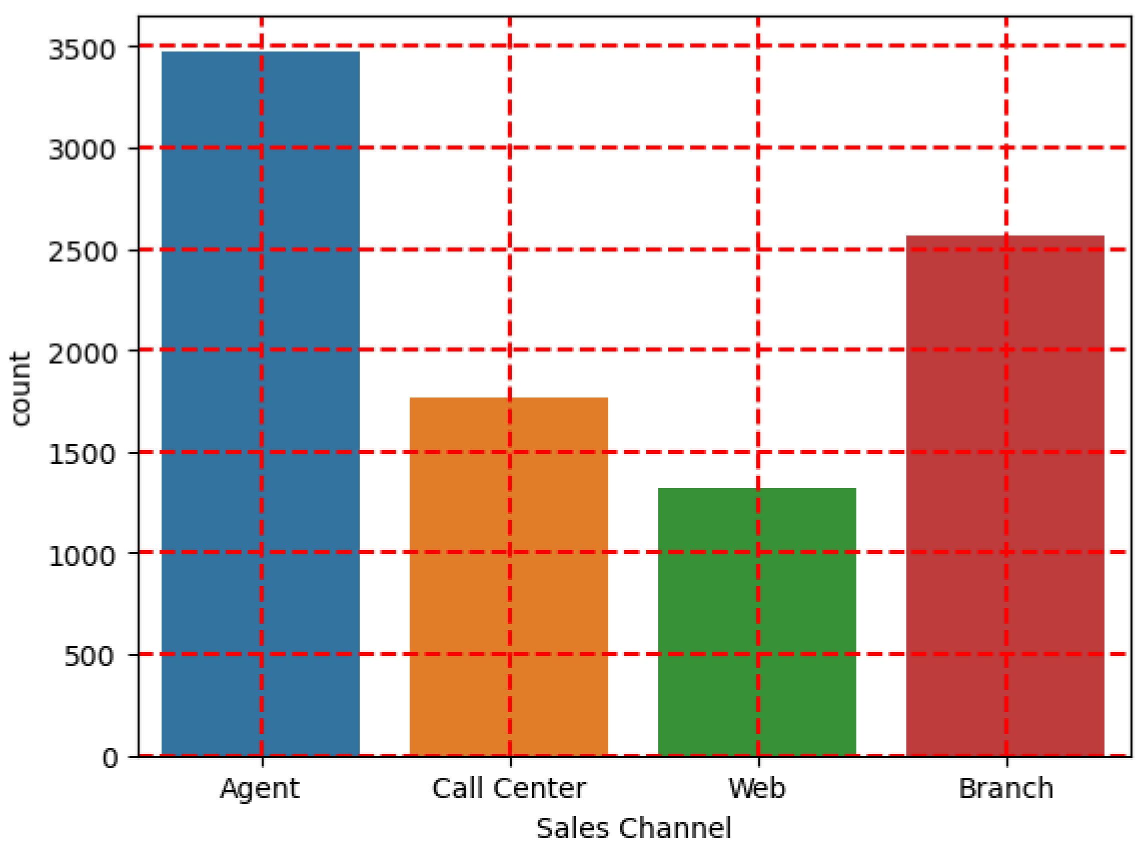
In [17]:
```
sns.countplot(x="Policy",data=data)
plt.xticks(rotation=90, ha="right")
plt.show()
```



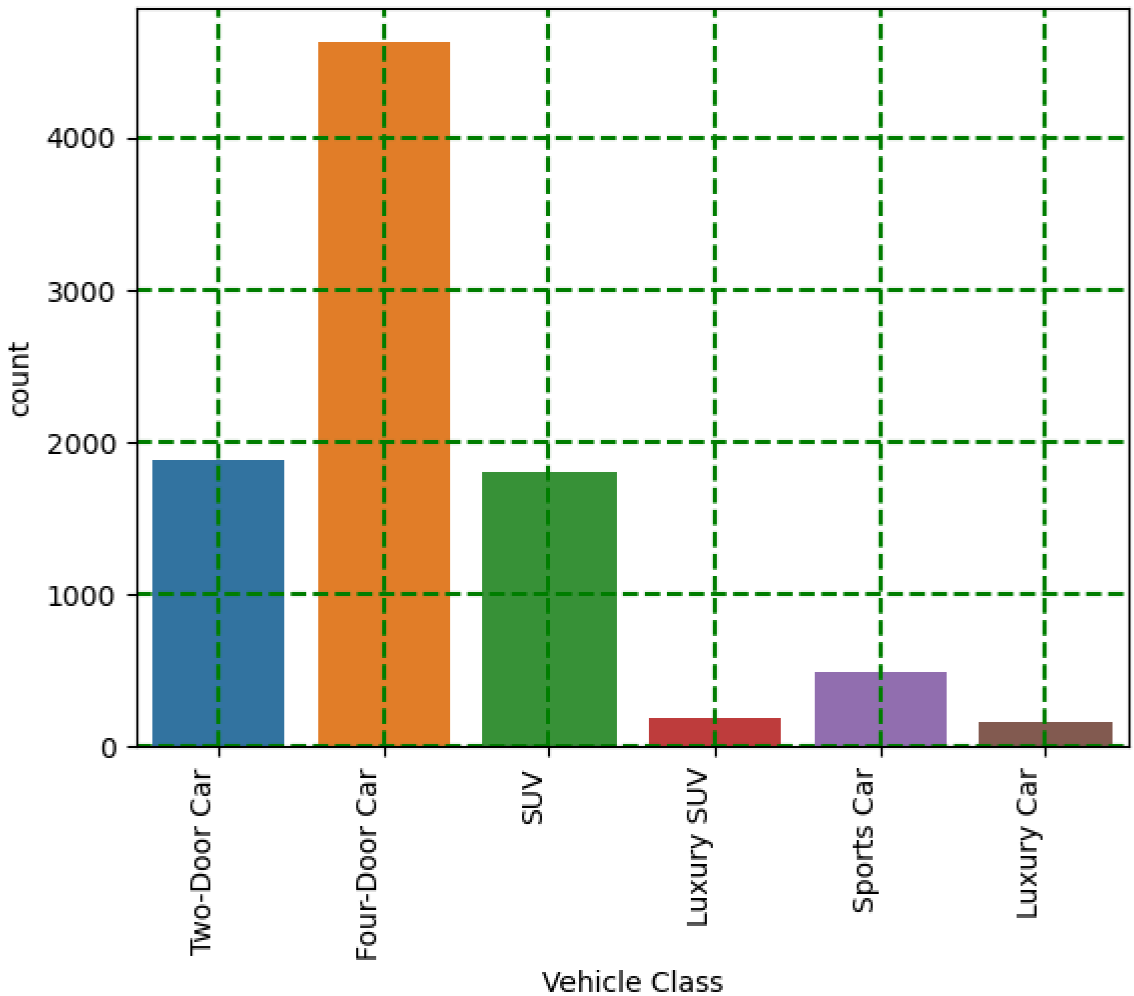
In [18]:
```
sns.countplot(x="Renew Offer Type",data=data)
plt.show()
```



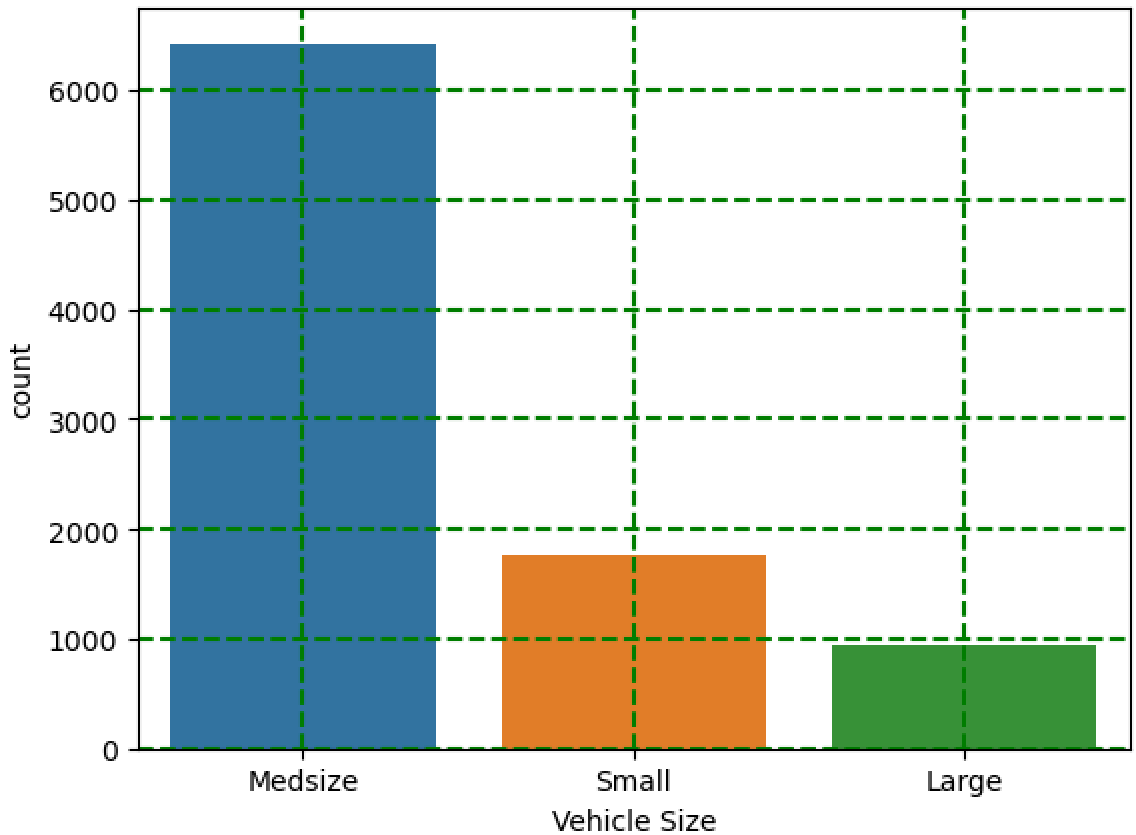
In [10]:
```
sns.countplot(x="Sales Channel",data=data)
plt.grid(color = 'r', linestyle = '--', linewidth = 1.5)
plt.show()
```



In [11]:
```
sns.countplot(x="Vehicle Class",data=data)
plt.grid(color = 'g', linestyle = '--', linewidth = 1.5)
plt.xticks(rotation=90, ha="right")
plt.show()
```

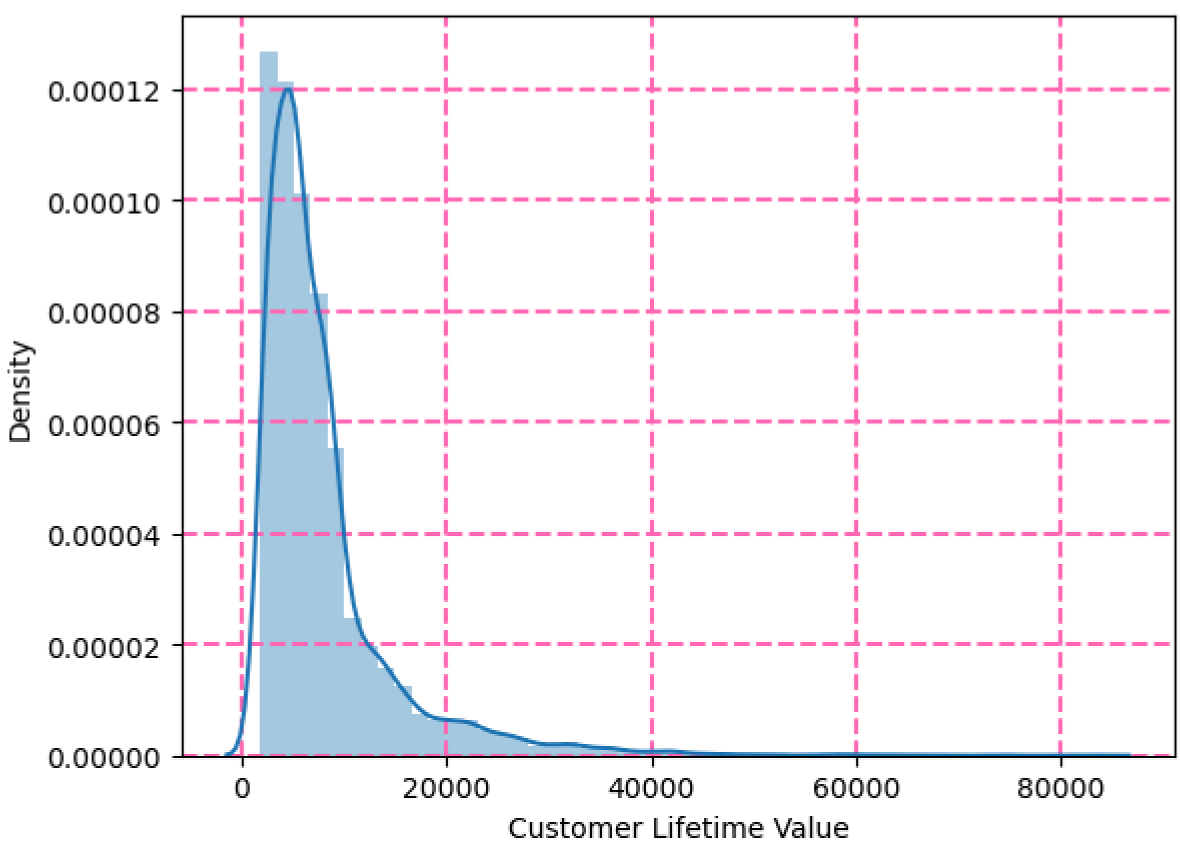


In [14]:
```
sns.countplot(x="Vehicle Size",data=data)
plt.grid(color = 'g', linestyle = '--', linewidth = 1.5)
plt.show()
```



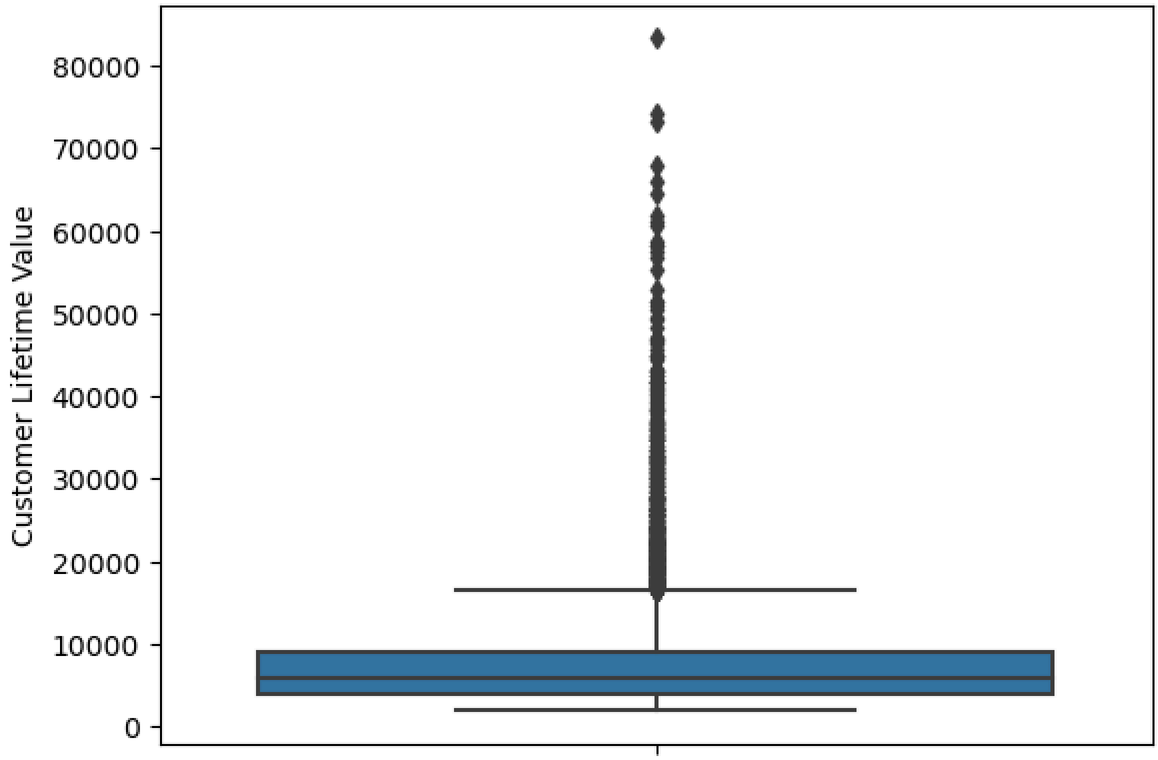
In [22]:

```
sns.distplot(data["Customer Lifetime Value"])
plt.grid(color = 'hotpink', linestyle = '--', linewidth = 1.5)
plt.show()
```

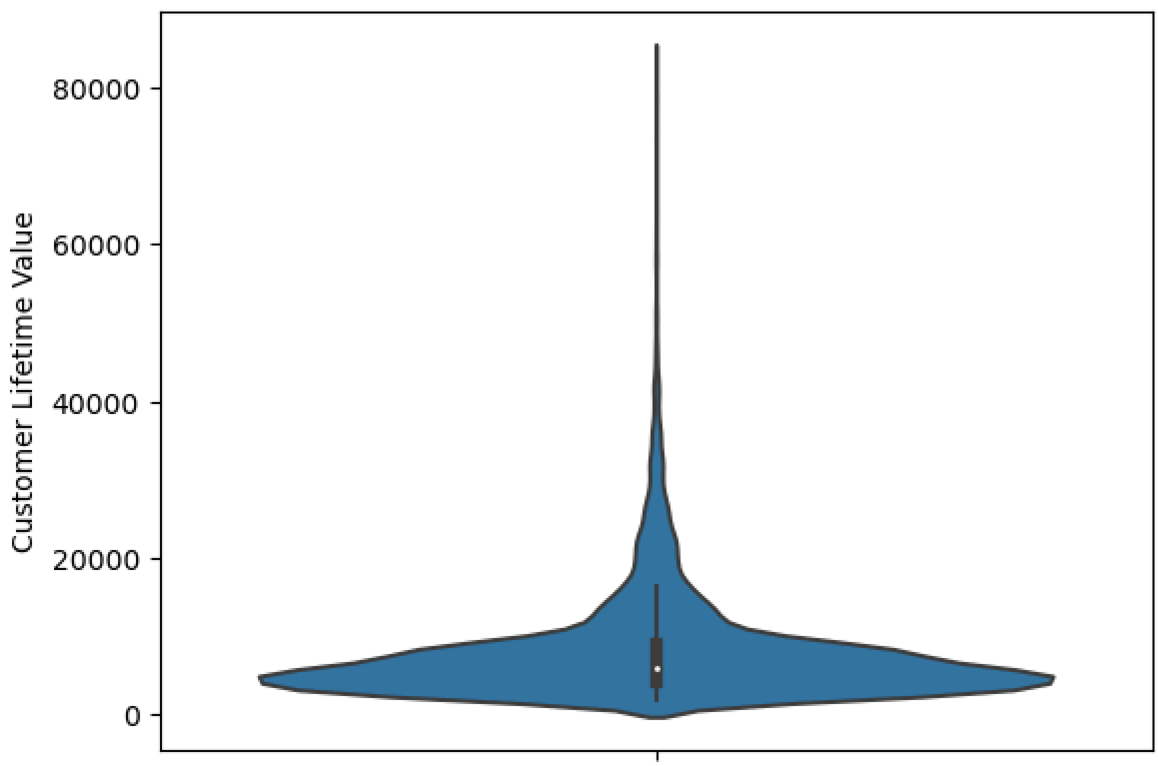


Loading [MathJax]/extensions/Safe.js

In [23]:
```
sns.boxplot(y="Customer Lifetime Value", data=data)
plt.show()
```



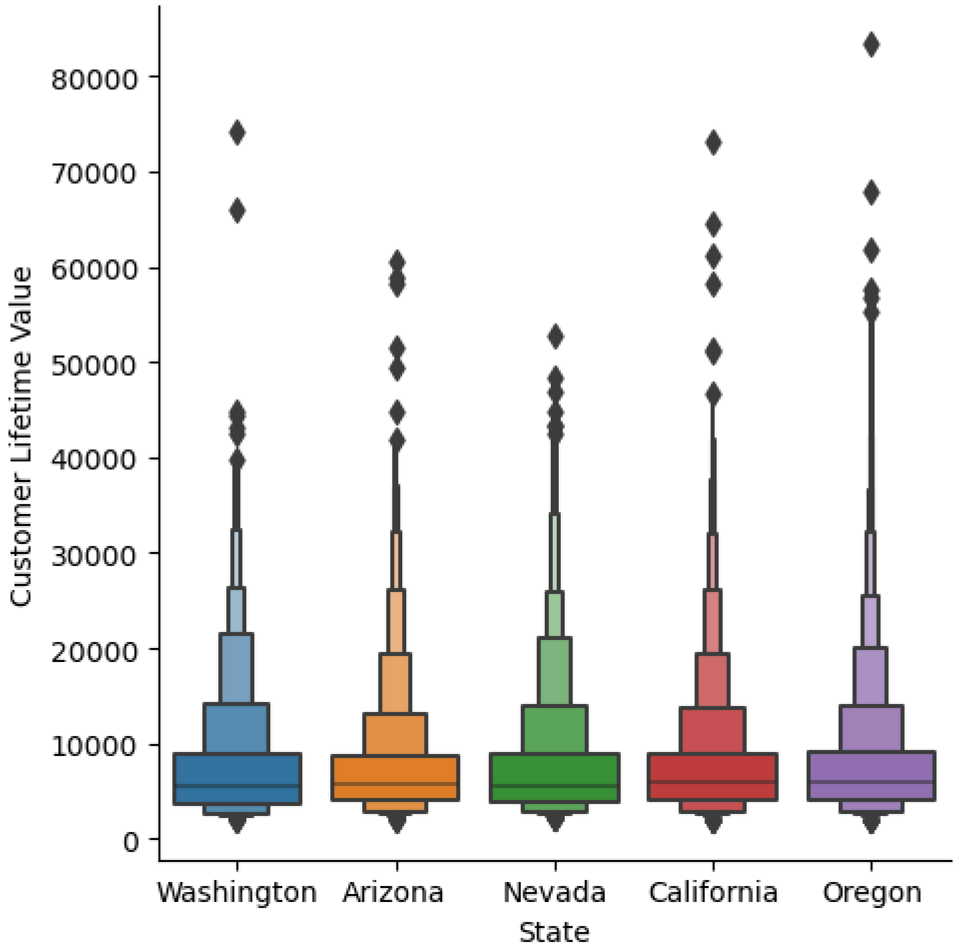
In [24]:
```
sns.violinplot(y="Customer Lifetime Value", data=data)
plt.show()
```



Loading [MathJax]/extensions/Safe.js

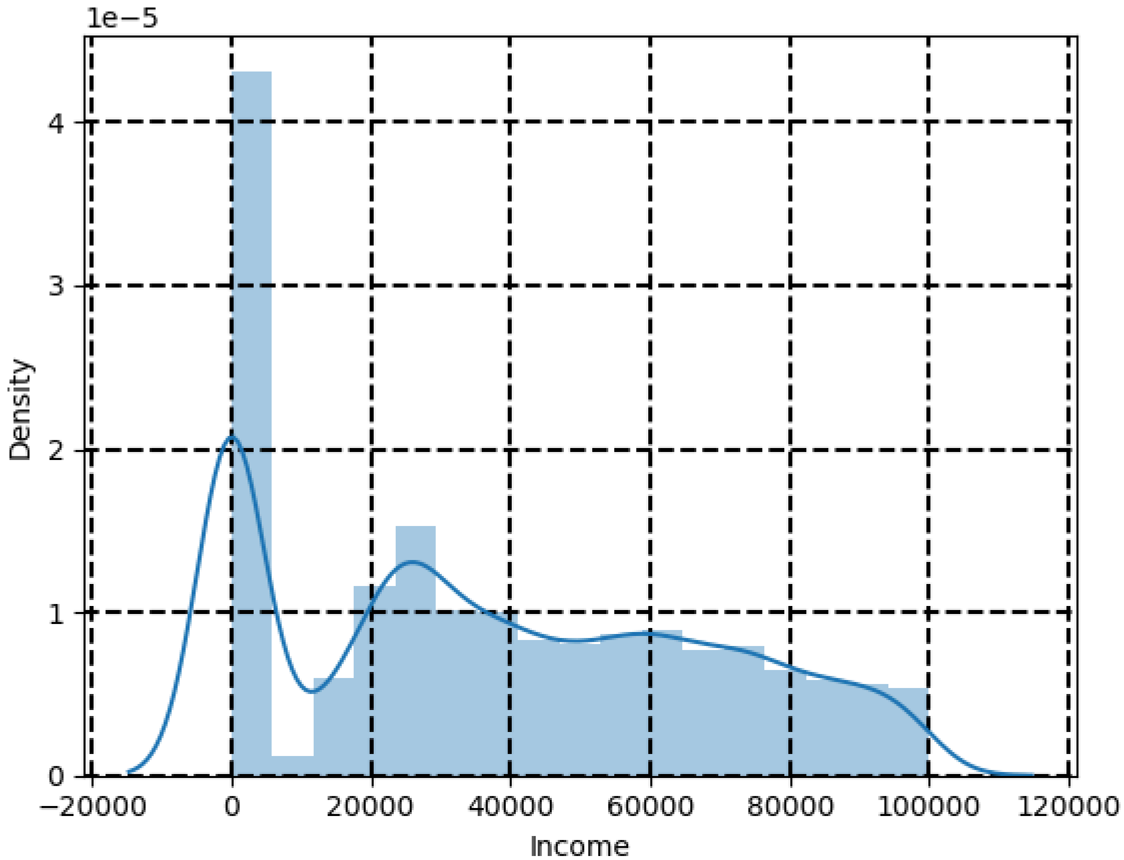
In [25]:
```
sns.catplot(data=data,
    x="State", y="Customer Lifetime Value", kind="boxen",)
```

Out[25]: <seaborn.axisgrid.FacetGrid at 0x1ec2a3dea40>

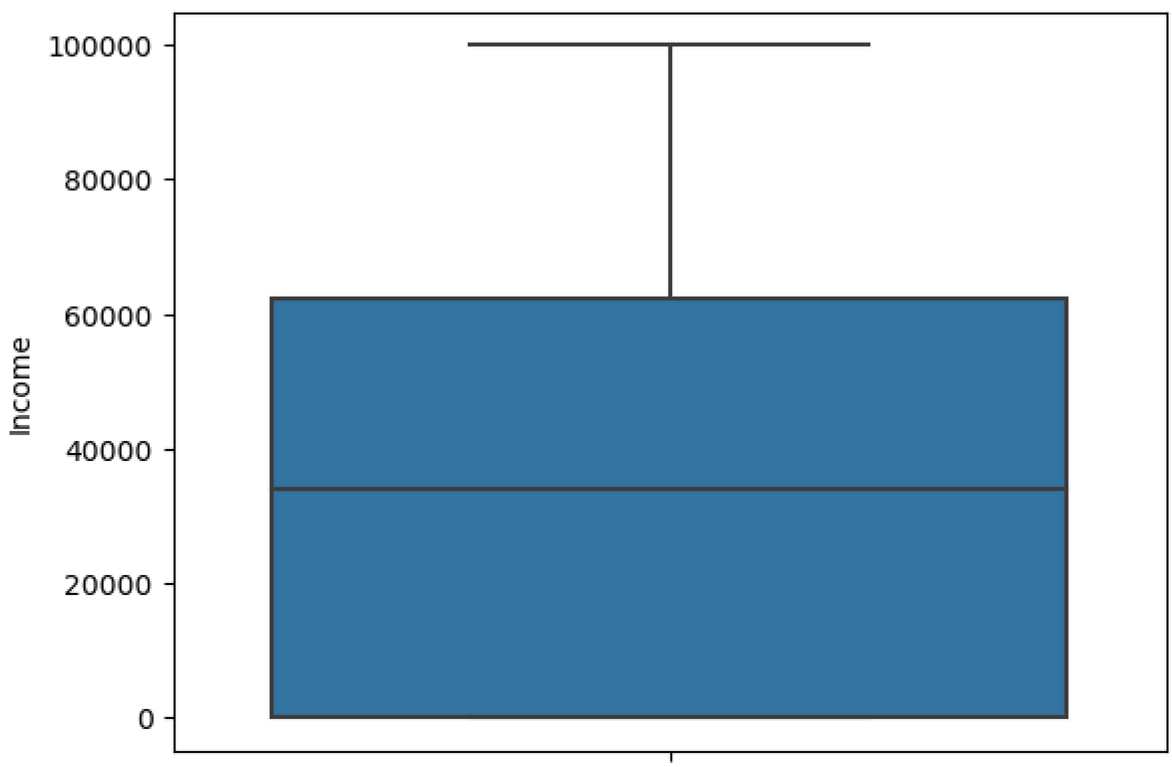


In [26]:
```
sns.distplot(data["Income"])
plt.grid(color = 'black', linestyle = '--', linewidth = 1.5)
plt.show()
```

Loading [MathJax]/extensions/Safe.js

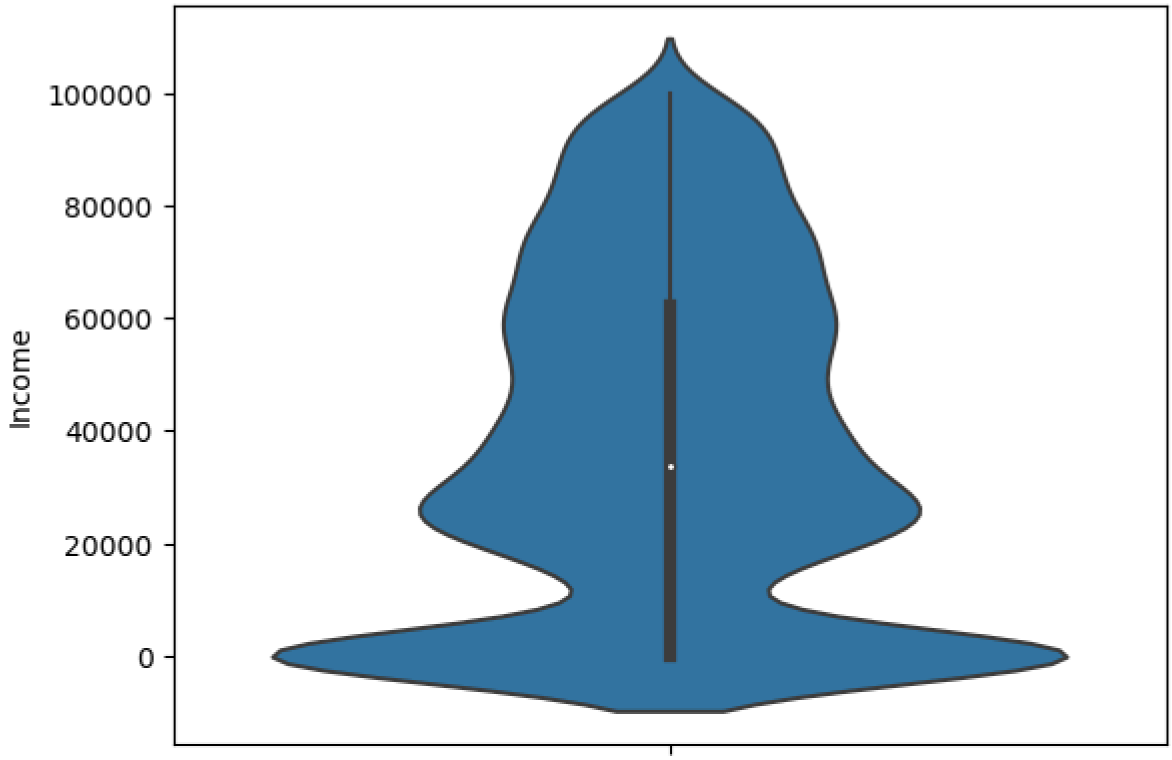


In [27]:
```
sns.boxplot(y="Income", data=data)
plt.show()
```



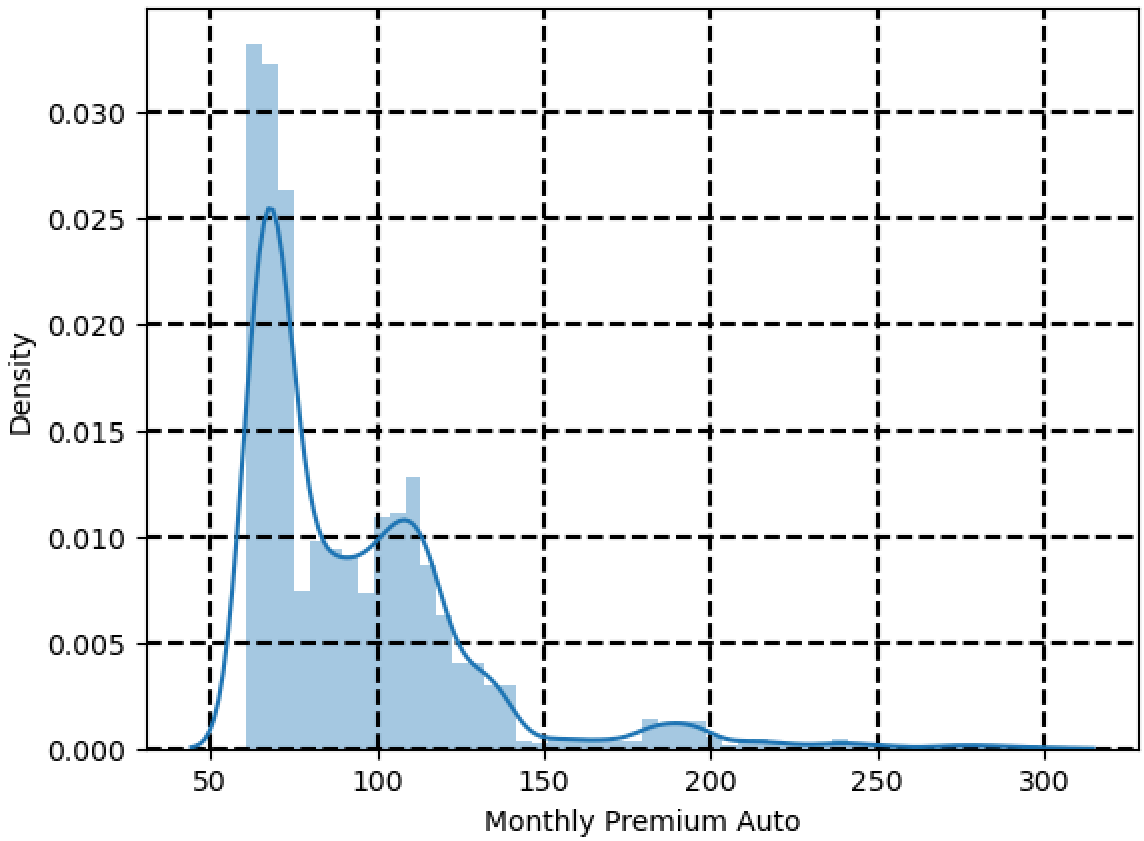
Loading [MathJax]/extensions/Safe.js

In [28]:
```
sns.violinplot(y="Income", data=data)
plt.show()
```

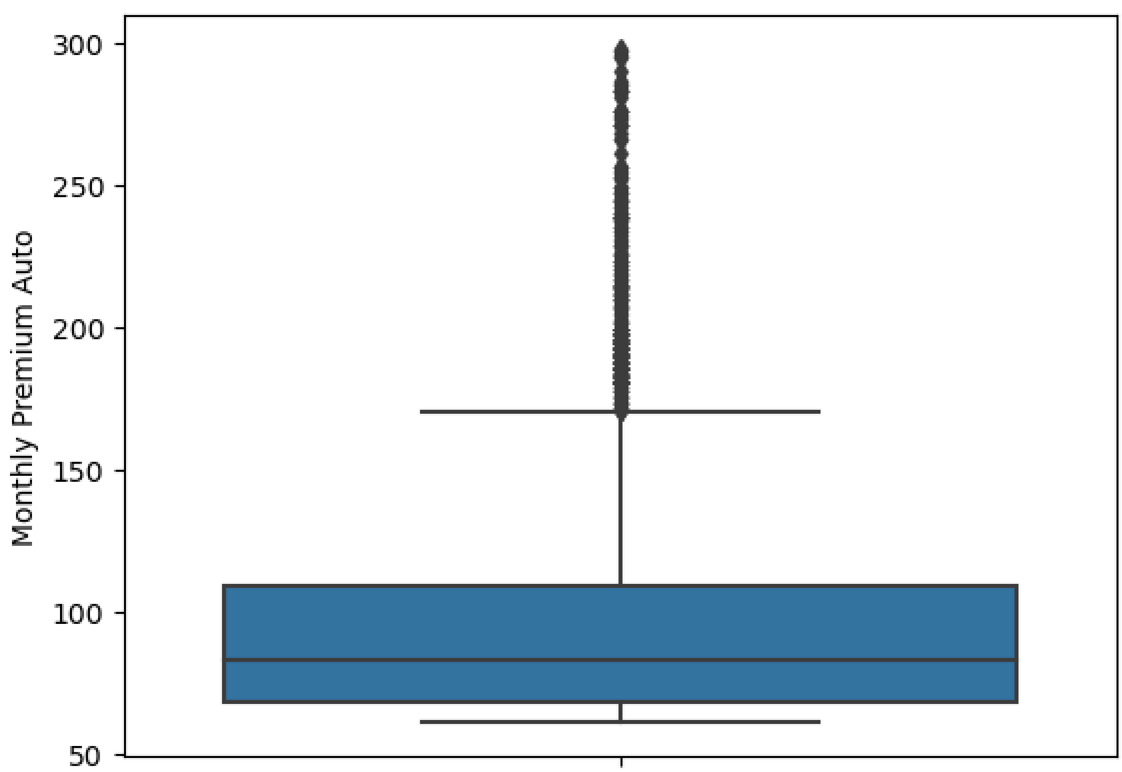


In [29]:
```
sns.distplot(data["Monthly Premium Auto"])
plt.grid(color = 'black', linestyle = '--', linewidth = 1.5)
plt.show()
```

Loading [MathJax]/extensions/Safe.js



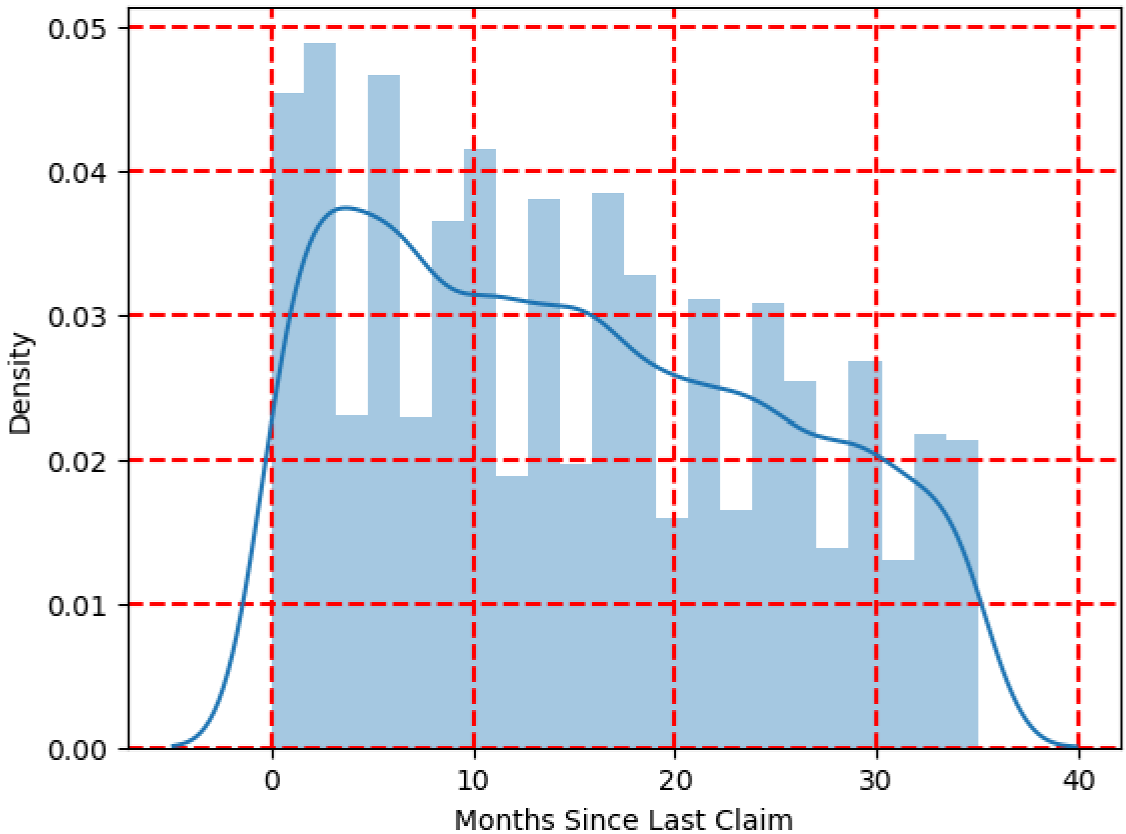
In [30]:
```
sns.boxplot(y="Monthly Premium Auto", data=data)
plt.show()
```



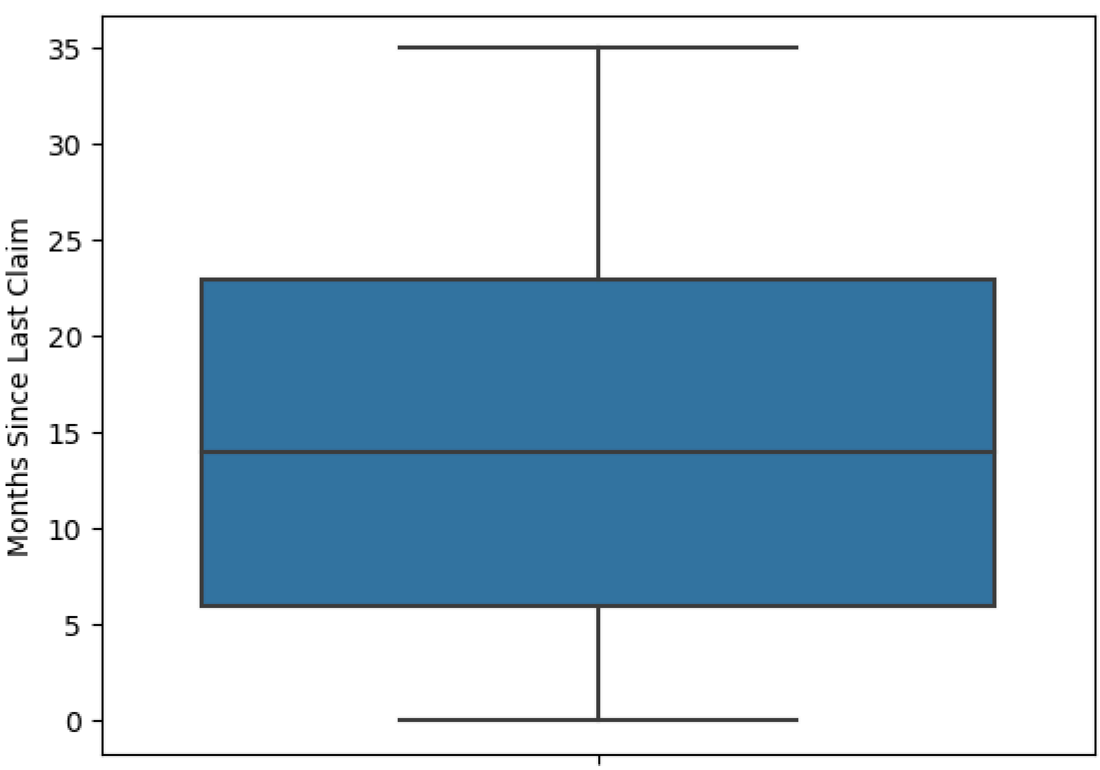
In [15]:
```
sns.distplot(data["Months Since Last Claim"])
plt.grid(color = 'r', linestyle = '--', linewidth = 1.5)
```

Loading [MathJax]/extensions/Safe.js

```
plt.show()
```

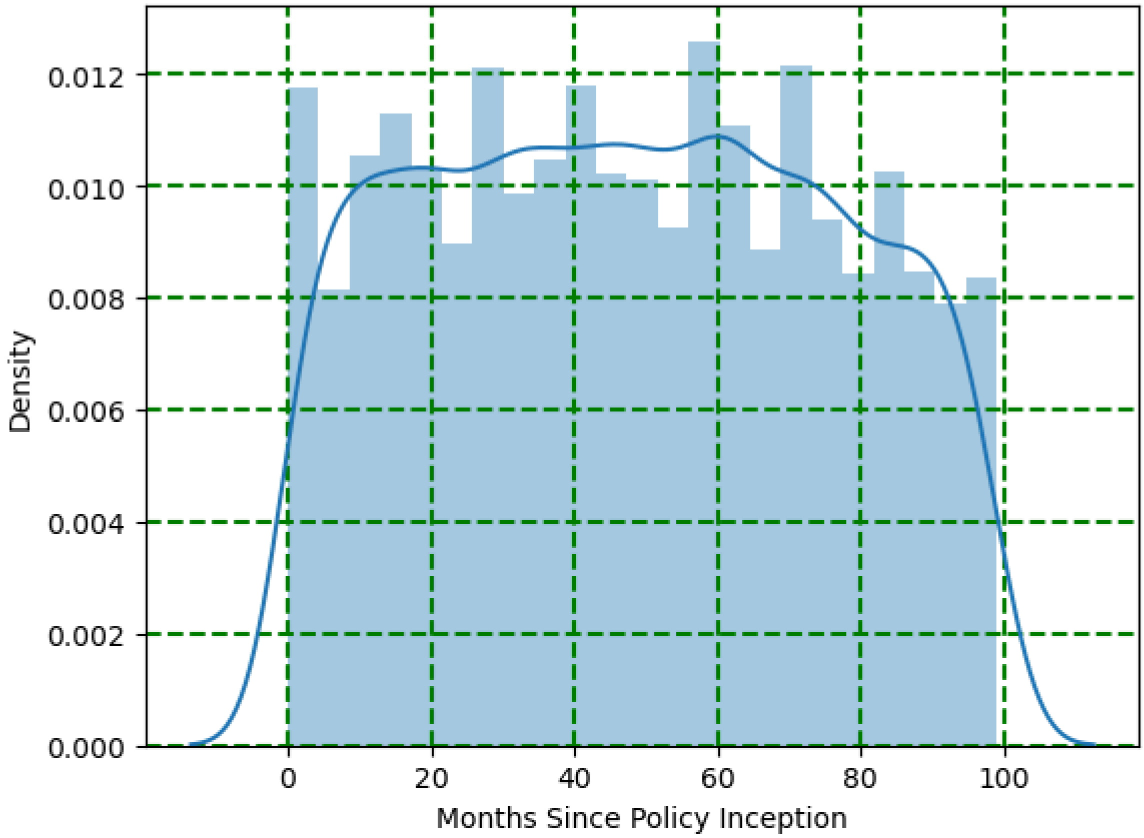


In [32]:
```
sns.boxplot(y="Months Since Last Claim", data=data)
plt.show()
```



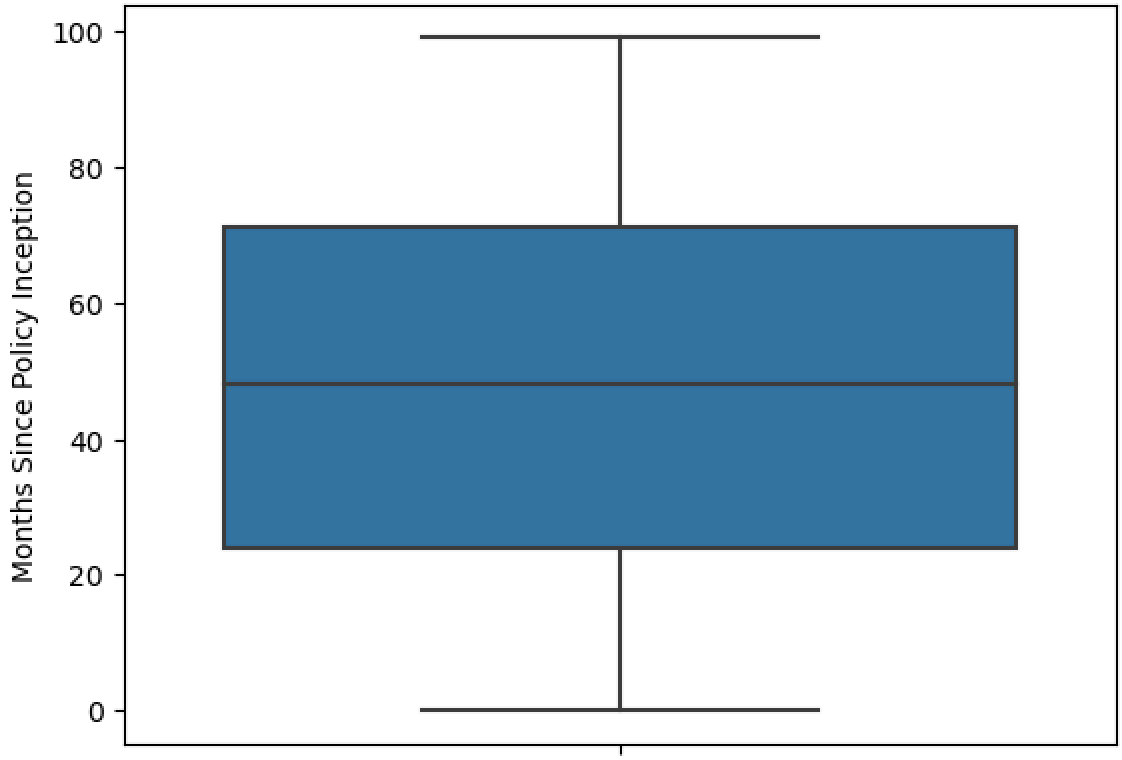
Loading [MathJax]/extensions/Safe.js

In [16]:
```
sns.distplot(data["Months Since Policy Inception"])
plt.grid(color = 'g', linestyle = '--', linewidth = 1.5)
plt.show()
```



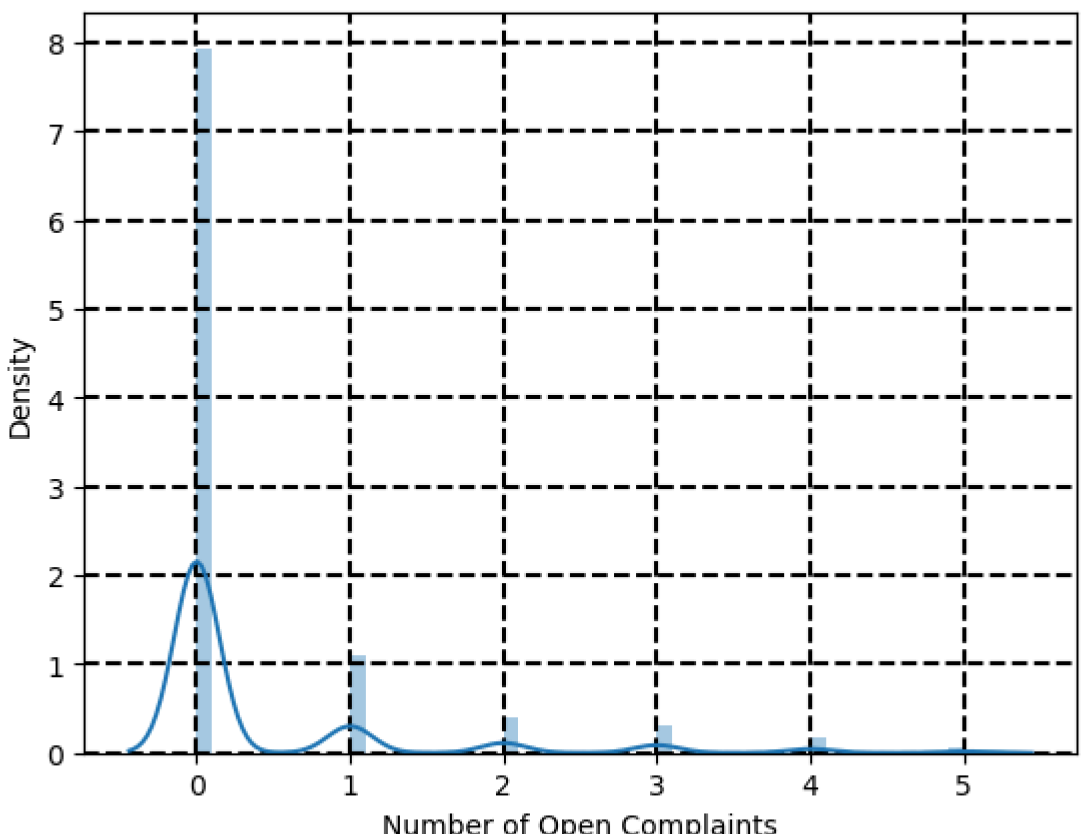
In [17]:
```
sns.boxplot(y="Months Since Policy Inception", data=data)
plt.show()
```

Loading [MathJax]/extensions/Safe.js



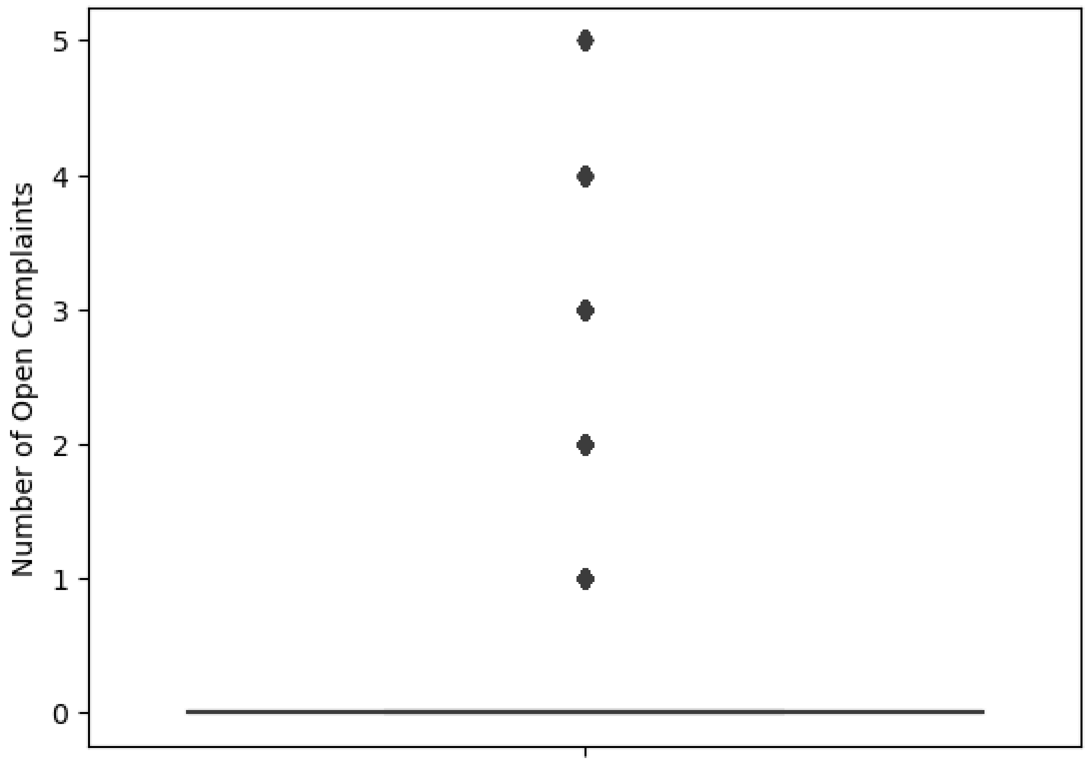
In [ ]:

In [18]:
```
sns.distplot(data["Number of Open Complaints"])
plt.grid(color = 'black', linestyle = '--', linewidth = 1.5)
plt.show()
```



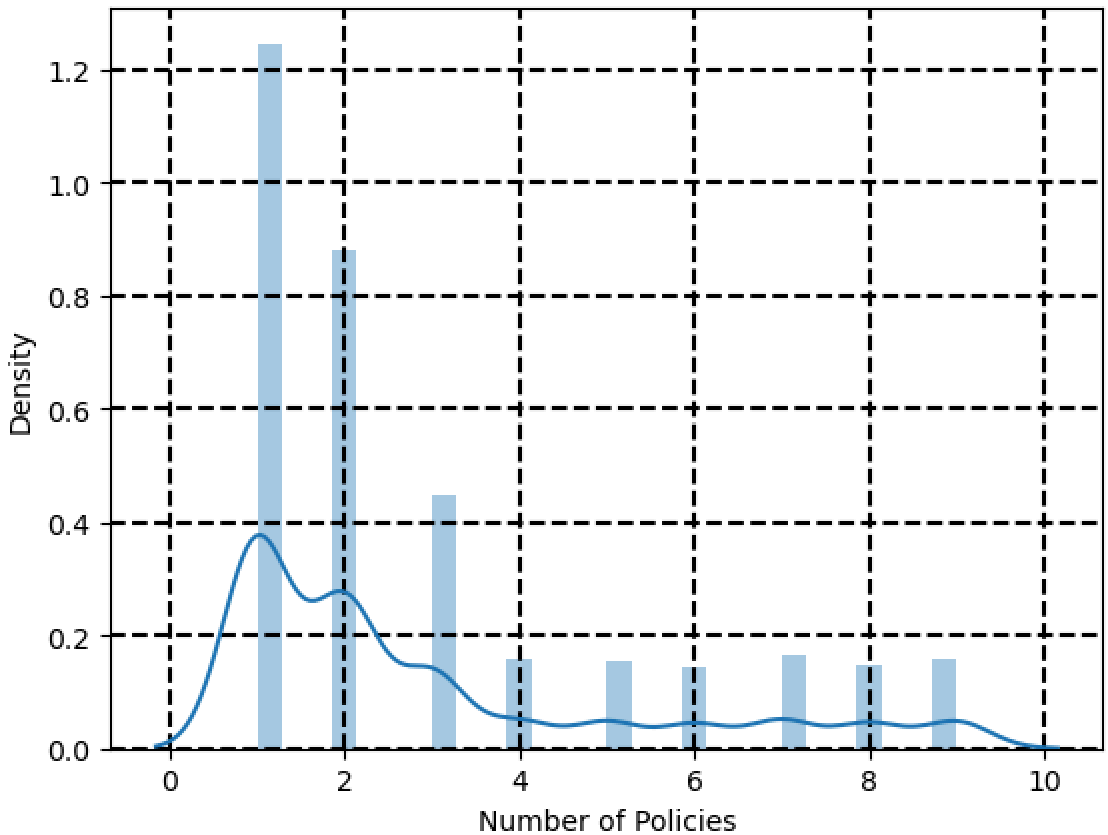
Loading [MathJax]/extensions/Safe.js

In [19]:
```
sns.boxplot(y="Number of Open Complaints", data=data)
plt.show()
```

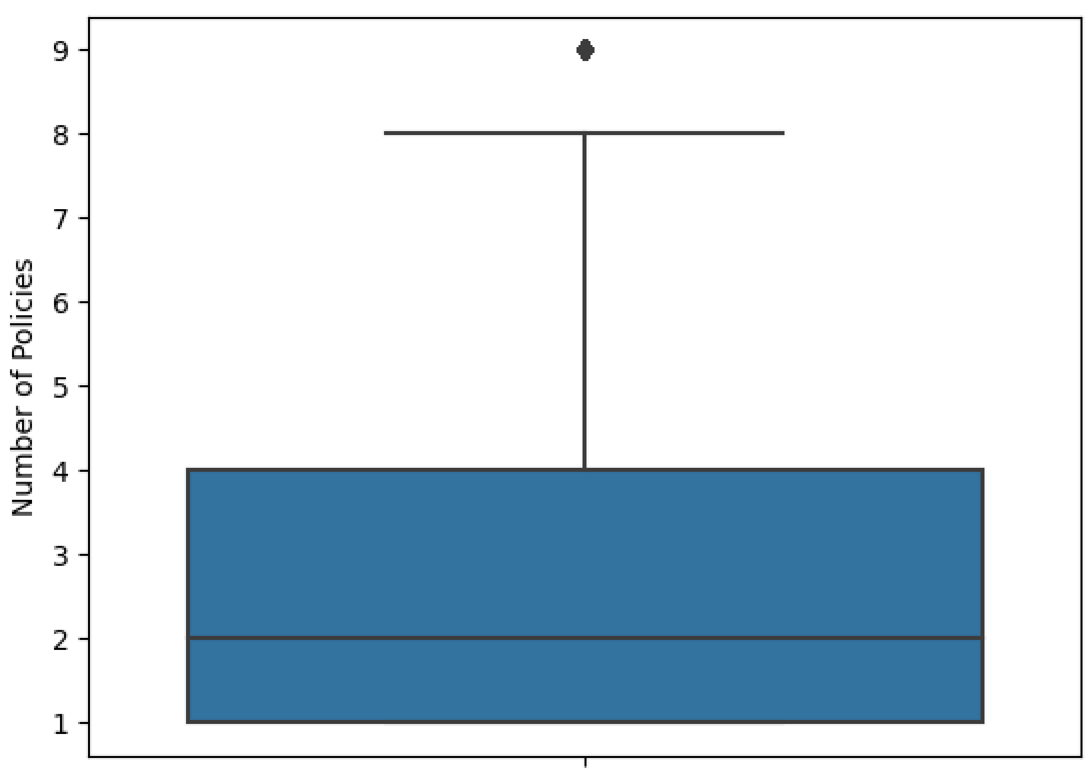


In [20]:
```
sns.distplot(data["Number of Policies"])
plt.grid(color = 'black', linestyle = '--', linewidth = 1.5)
plt.show()
```

Loading [MathJax]/extensions/Safe.js



In [21]:
```
sns.boxplot(y="Number of Policies", data=data)
plt.show()
```

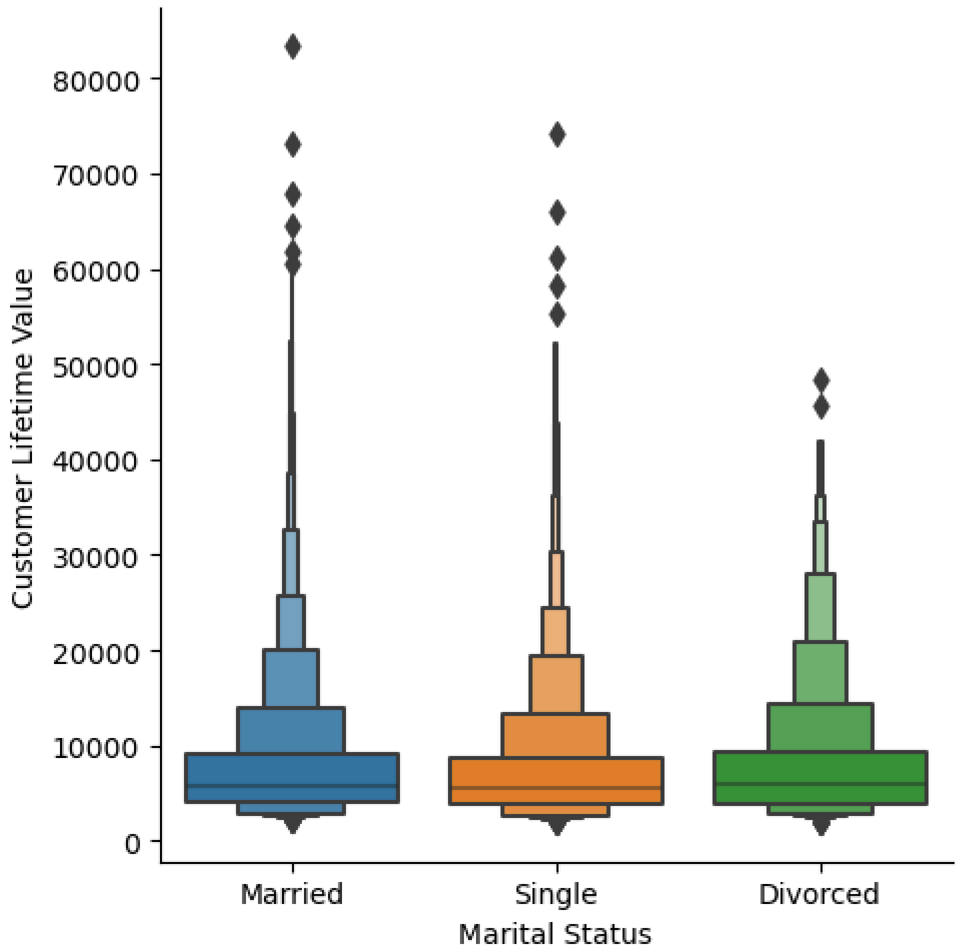


In [22]:
```
sns.catplot(data=data,
    x="Marital Status", y="Customer Lifetime Value", kind="boxen",
```

Loading [MathJax]/extensions/Safe.js

```
)
```

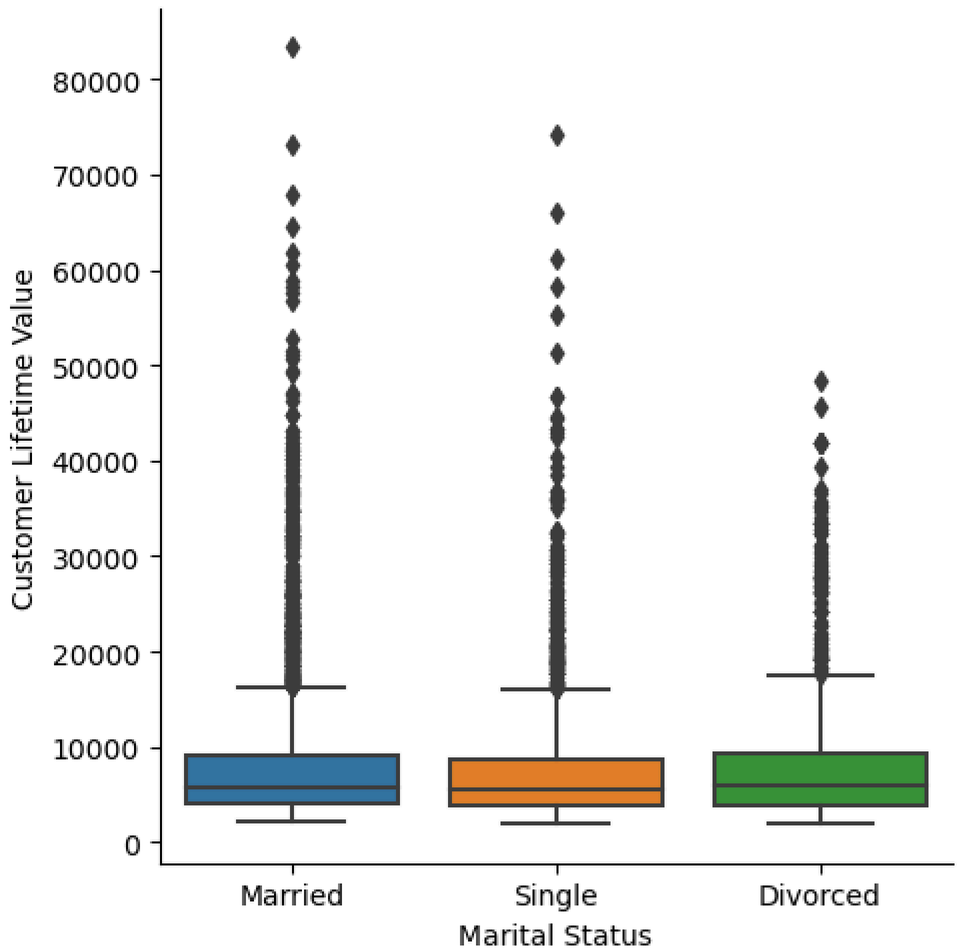
Out[22]: <seaborn.axisgrid.FacetGrid at 0x2b420ee79d0>



In [23]:
```
sns.catplot(data=data,
    x="Marital Status", y="Customer Lifetime Value", kind="box",)
```

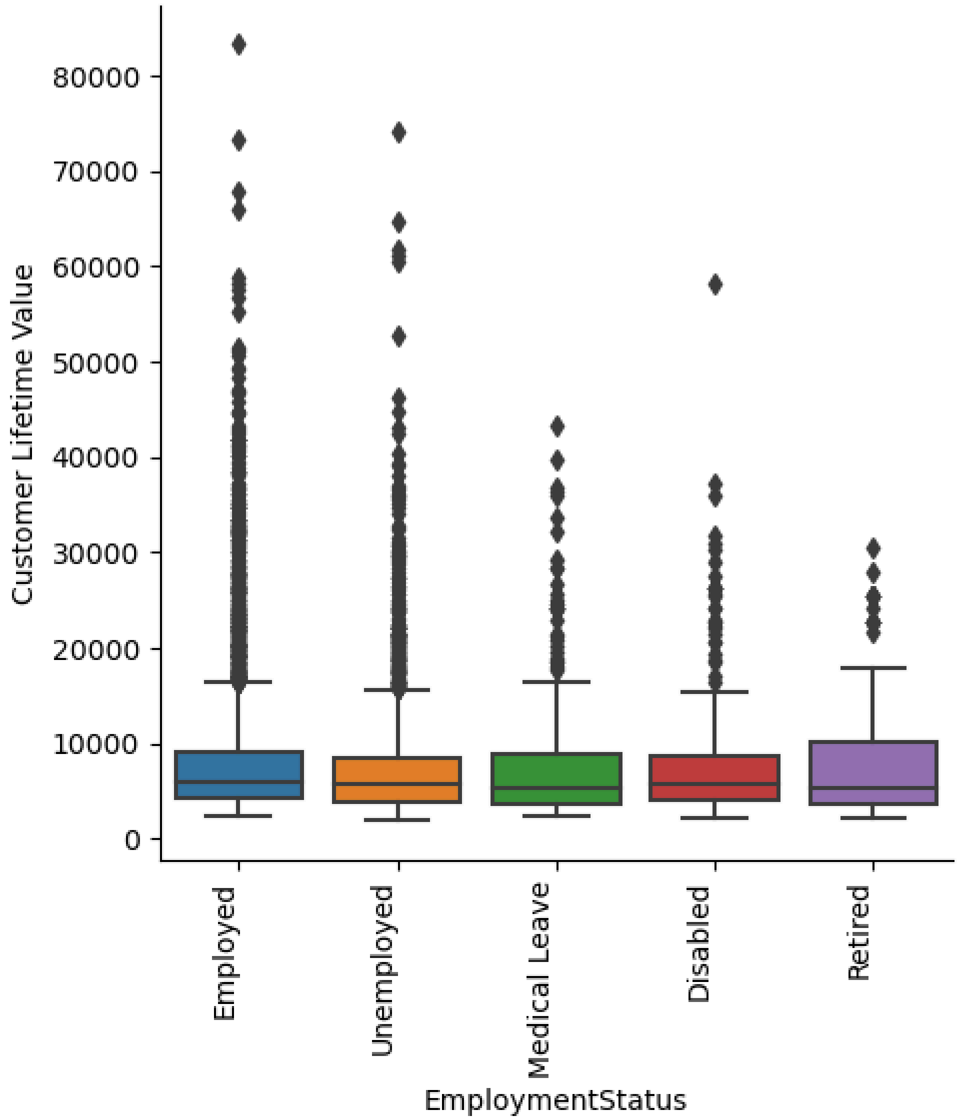
Out[23]: <seaborn.axisgrid.FacetGrid at 0x2b41f3f8850>

Loading [MathJax]/extensions/Safe.js



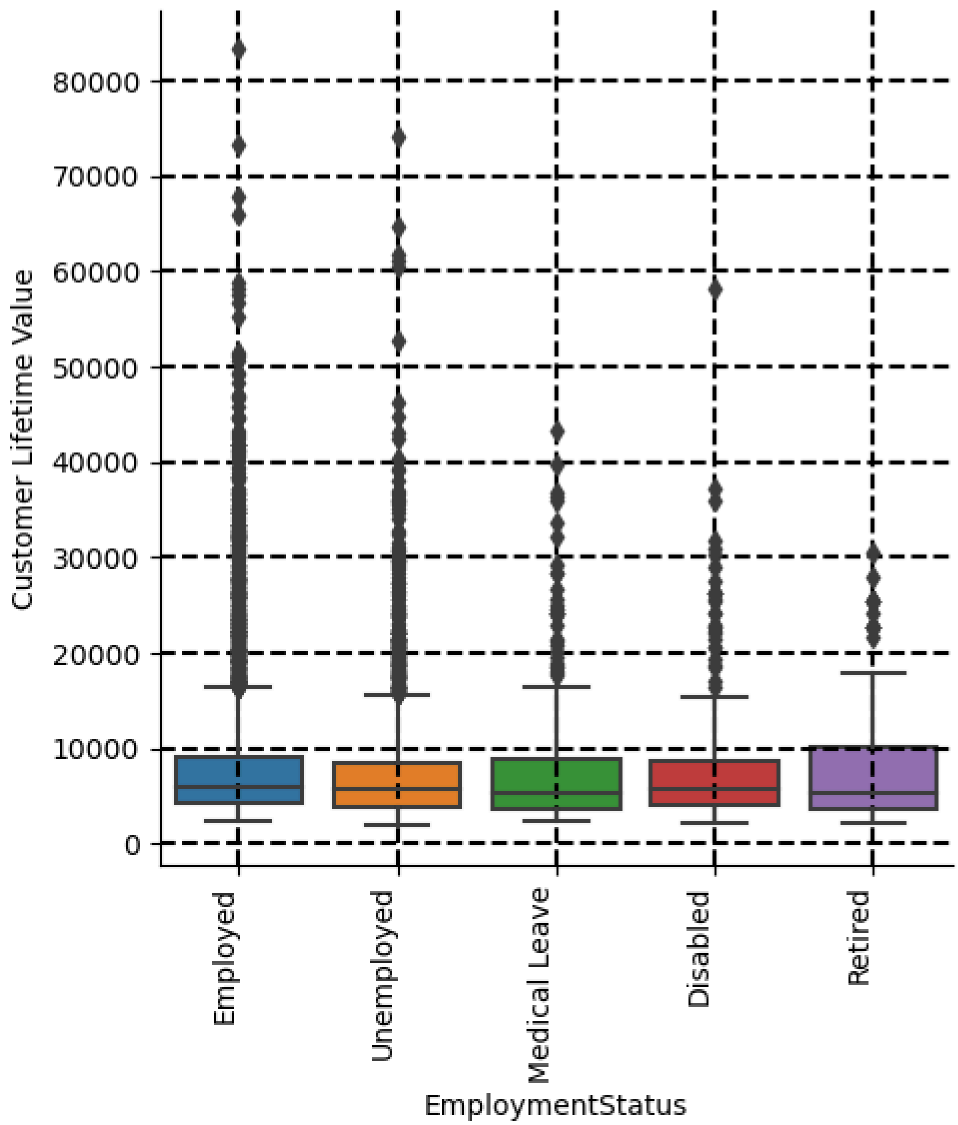
In [24]:
```
sns.catplot(data=data,
    x="EmploymentStatus", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

Out[24]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Employed'),
  Text(1, 0, 'Unemployed'),
  Text(2, 0, 'Medical Leave'),
  Text(3, 0, 'Disabled'),
  Text(4, 0, 'Retired')])
```



In [25]:

```
sns.catplot(data=data,
    x="EmploymentStatus", y="Customer Lifetime Value", kind="box",
)
plt.grid(color = 'black', linestyle = '--', linewidth = 1.5)
plt.xticks(rotation=90, ha="right")
plt.show()
```



Customer 9134 non-null object 1 State 9134 non-null object 2 Customer Lifetime Value 9134 non-null float64 3 Response 9134 non-null object 4 Coverage 9134 non-null object 5 Education 9134 non-null object 6 Effective To Date 9134 non-null object 7 EmploymentStatus 9134 non-null object 8 Gender 9134 non-null object 9 Income 9134 non-null int64

10 Location Code 9134 non-null object 11 Marital Status 9134 non-null object 12 Monthly Premium Auto 9134 non-null int64

13 Months Since Last Claim 9134 non-null int64

14 Months Since Policy Inception 9134 non-null int64

15 Number of Open Complaints 9134 non-null int64

16 Number of Policies 9134 non-null int64

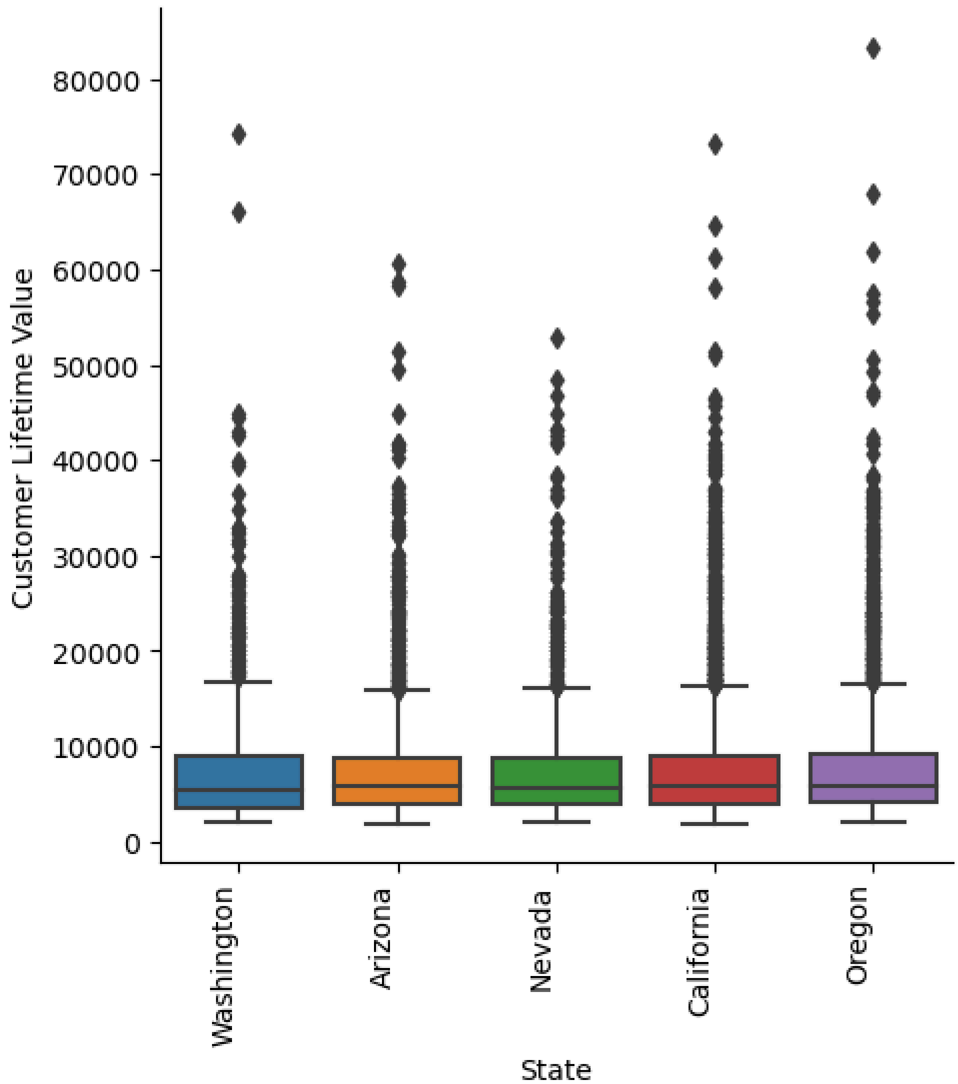
17 Policy Type 9134 non-null object 18 Policy 9134 non-null object 19 Renew Offer Type 9134 non-null object 20 Sales Channel 9134 non-null object 21 Total Claim Amount 9134 non-null float64 22 Vehicle Class 9134 non-null object 23 Vehicle Size 9134 non-null object

In [43]:
```
sns.catplot(data=data,
            , y="Customer Lifetime Value", kind="box",
```

Loading [MathJax]/extensions/Safe.js

```
)
plt.xticks(rotation=90, ha="right")
```

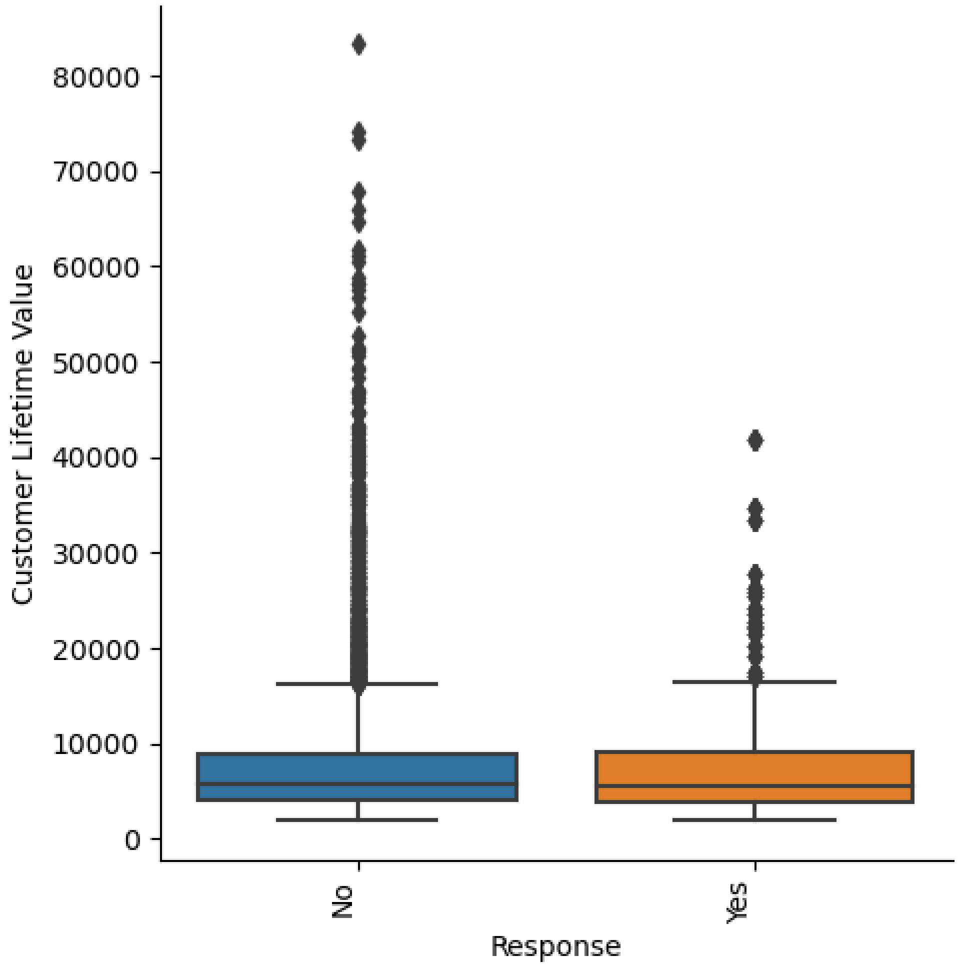
Out[43]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Washington'),
  Text(1, 0, 'Arizona'),
  Text(2, 0, 'Nevada'),
  Text(3, 0, 'California'),
  Text(4, 0, 'Oregon')])
```



In [44]:
```
sns.catplot(data=data,
    x="Response", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

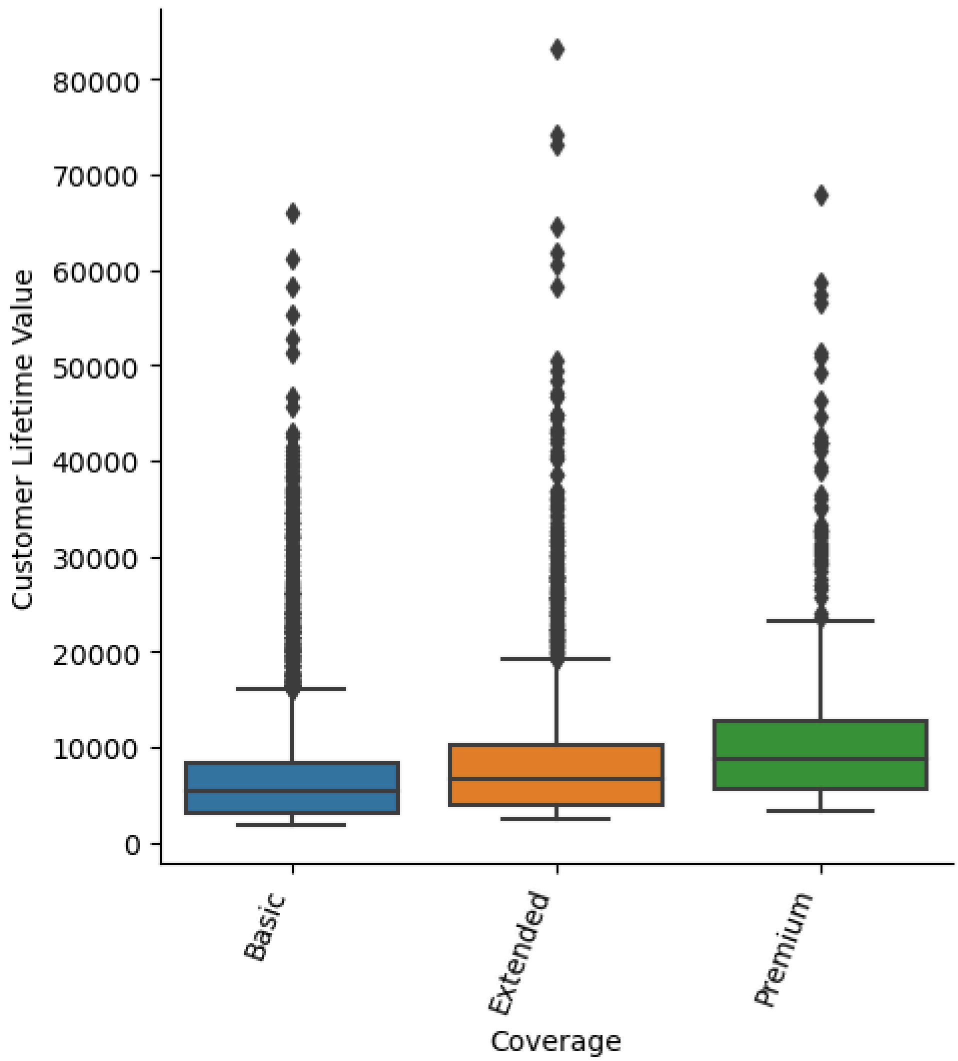
Out[44]:
```
(array([0, 1]), [Text(0, 0, 'No'), Text(1, 0, 'Yes')])
```

Loading [MathJax]/extensions/Safe.js



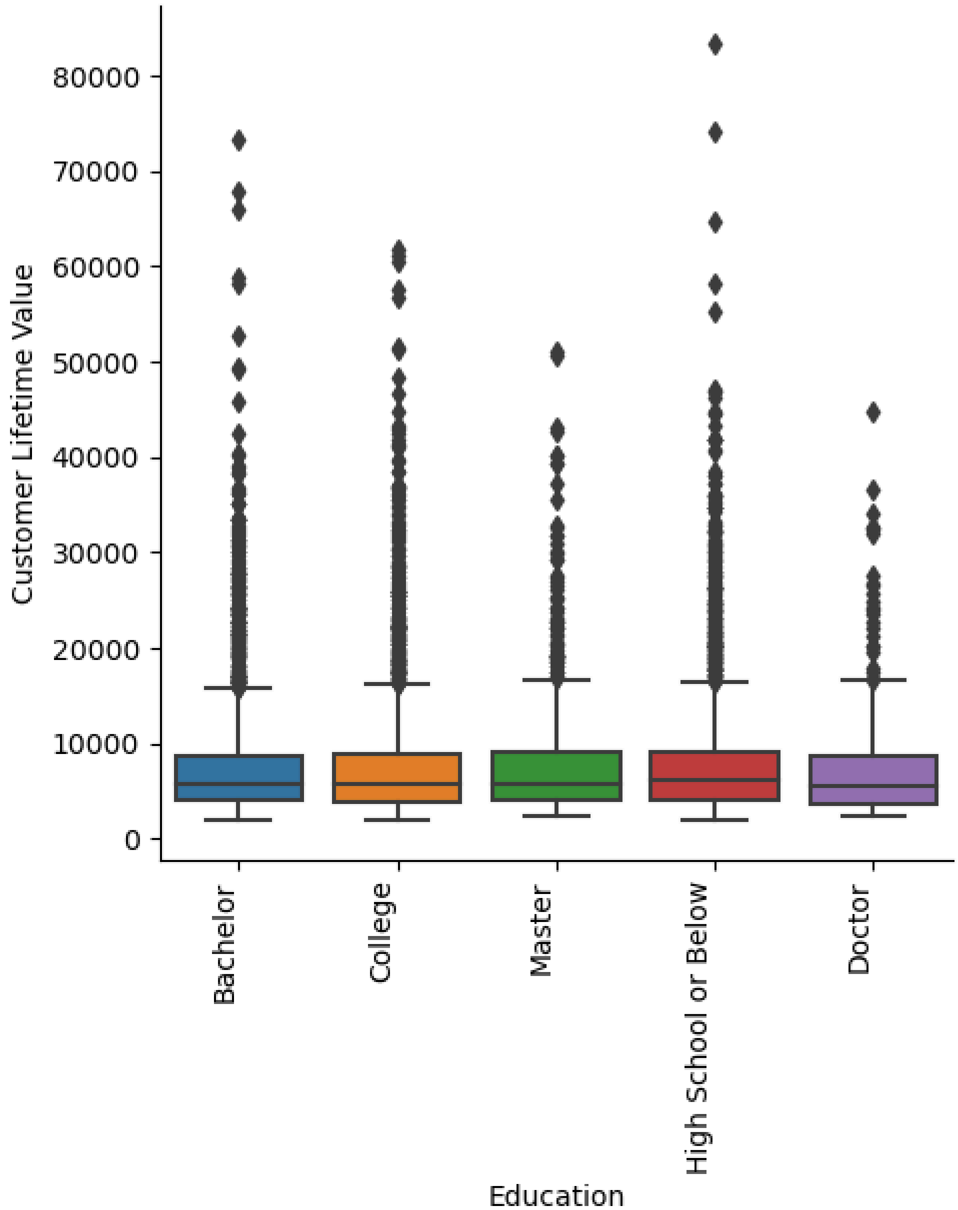
In [26]:
```
sns.catplot(data=data,
    x="Coverage", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=70, ha="right")
```

Out[26]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'Basic'), Text(1, 0, 'Extended'), Text(2, 0, 'Premium')])
```



In [27]:
```
sns.catplot(data=data,
    x="Education", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

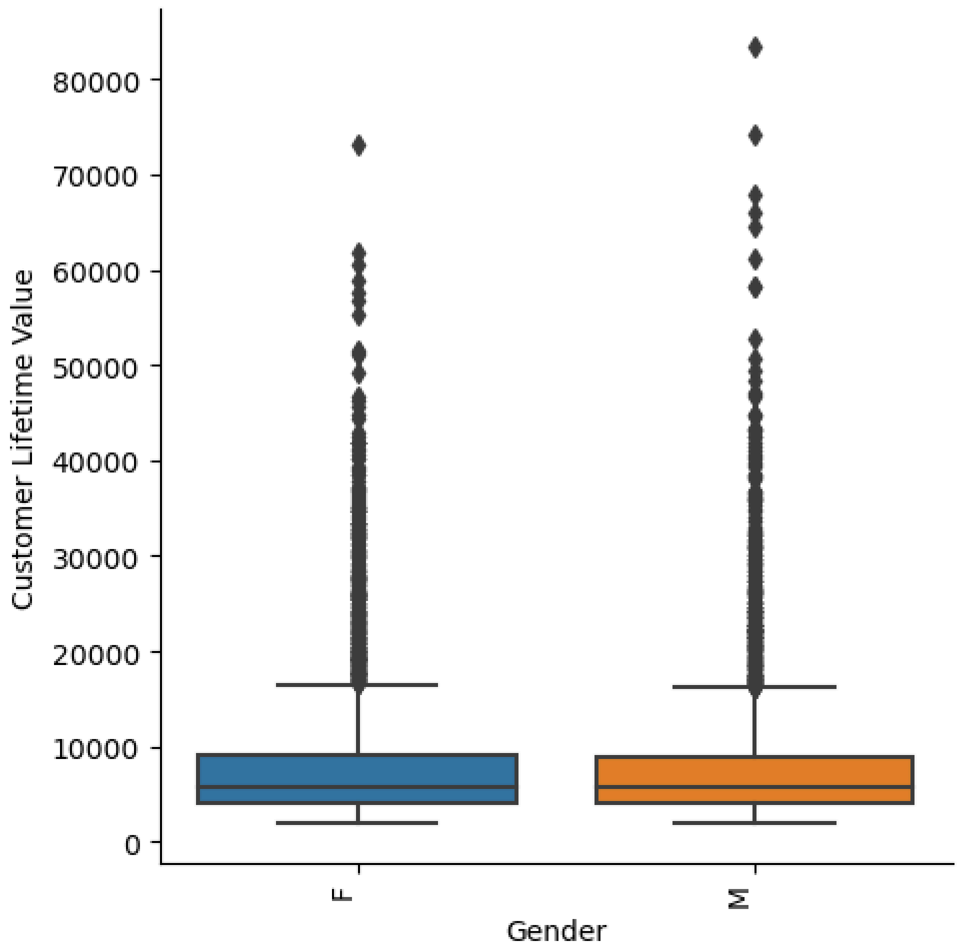
Out[27]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Bachelor'),
  Text(1, 0, 'College'),
  Text(2, 0, 'Master'),
  Text(3, 0, 'High School or Below'),
  Text(4, 0, 'Doctor')])
```



In [28]:
```
sns.catplot(data=data,
    x="Gender", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

Out[28]: (array([0, 1]), [Text(0, 0, 'F'), Text(1, 0, 'M')])

Loading [MathJax]/extensions/Safe.js

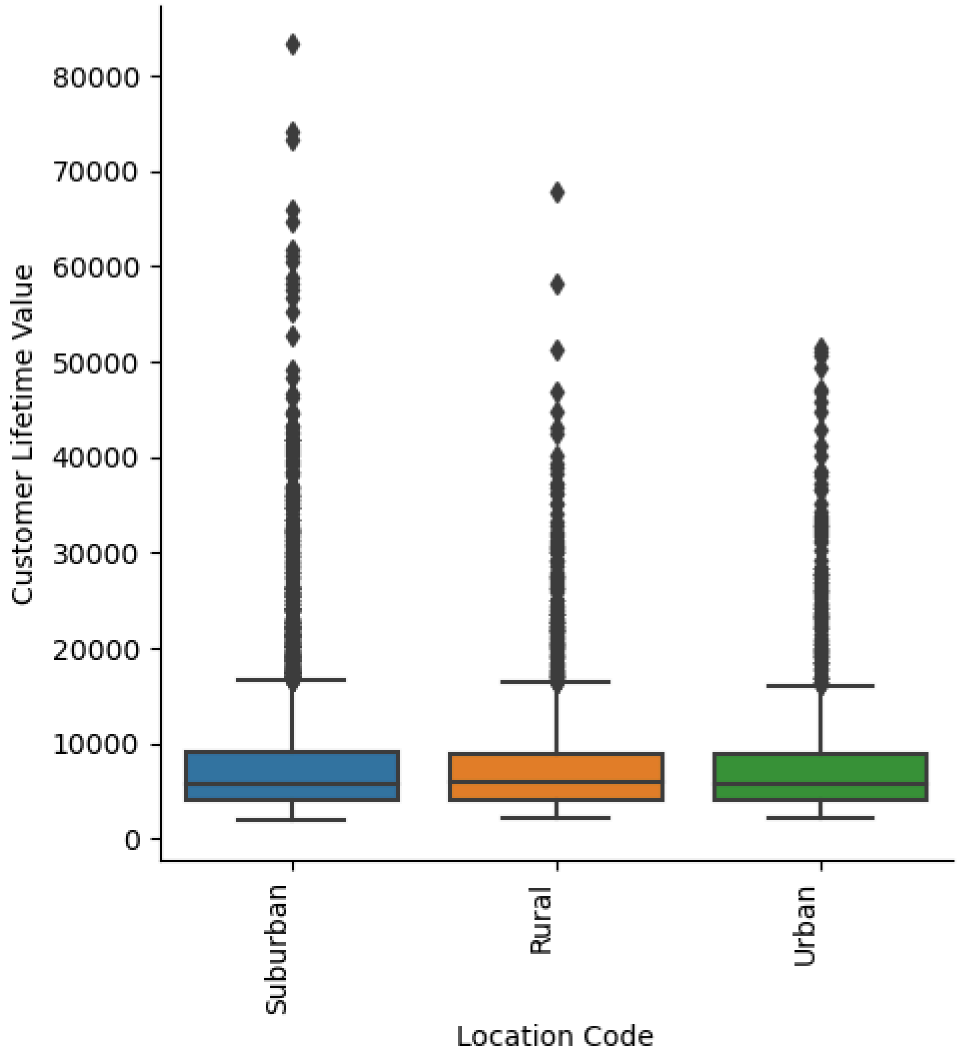


In [29]:

```
sns.catplot(data=data,
    x="Location Code", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

Out[29]:

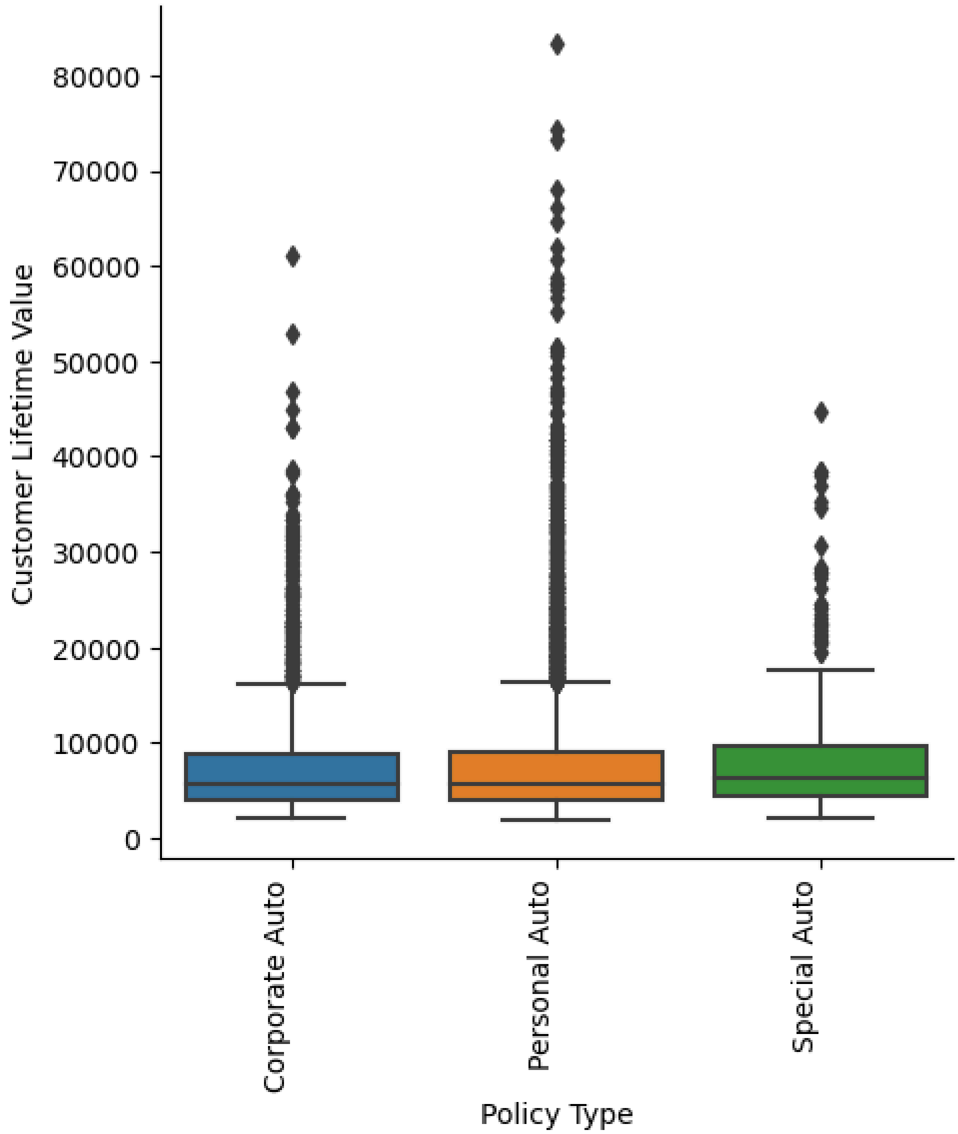
```
(array([0, 1, 2]),
 [Text(0, 0, 'Suburban'), Text(1, 0, 'Rural'), Text(2, 0, 'Urban')])
```




```
In [30]: sns.catplot(data=data,
             x="Policy Type", y="Customer Lifetime Value", kind="box",)
         plt.xticks(rotation=90, ha="right")
```

```
Out[30]: (array([0, 1, 2]),
          [Text(0, 0, 'Corporate Auto'),
           Text(1, 0, 'Personal Auto'),
           Text(2, 0, 'Special Auto')])
```

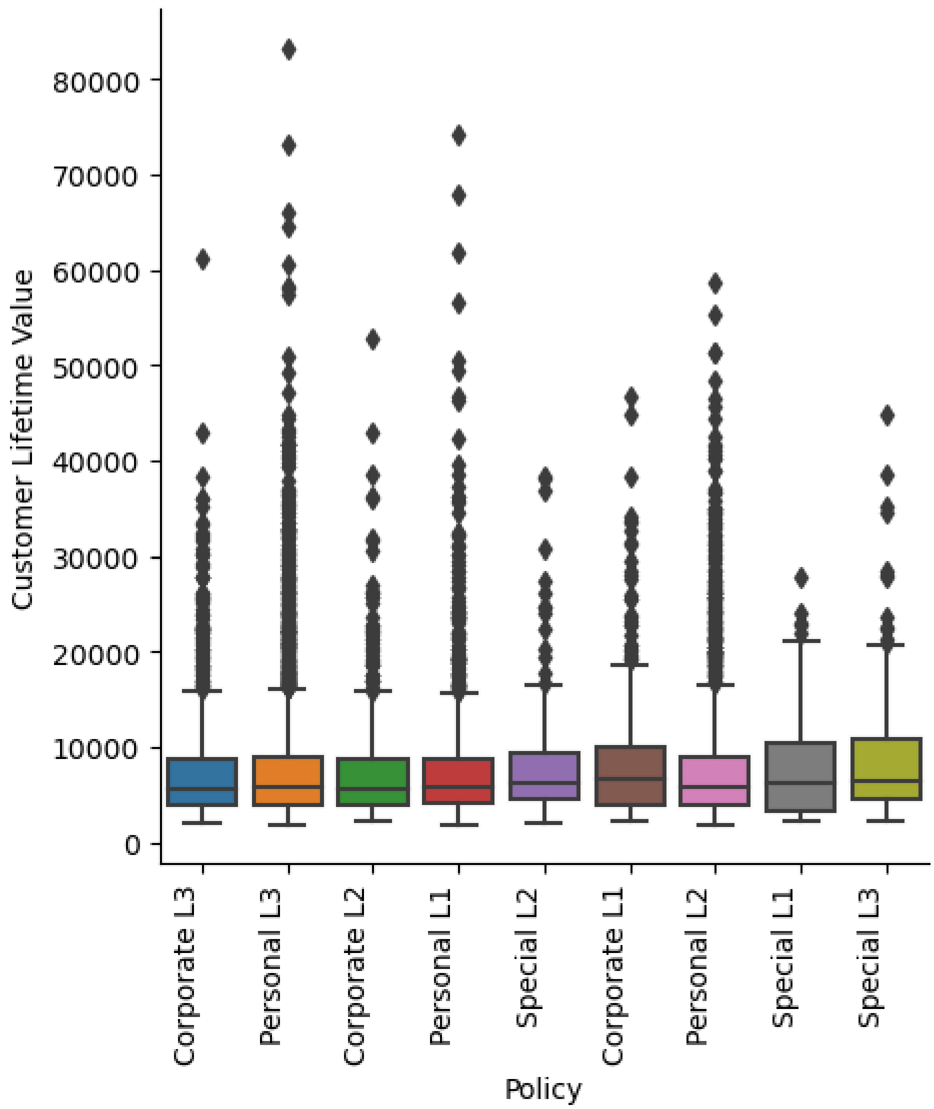
Loading [MathJax]/extensions/Safe.js



In [31]:
```
sns.catplot(data=data,
    x="Policy", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

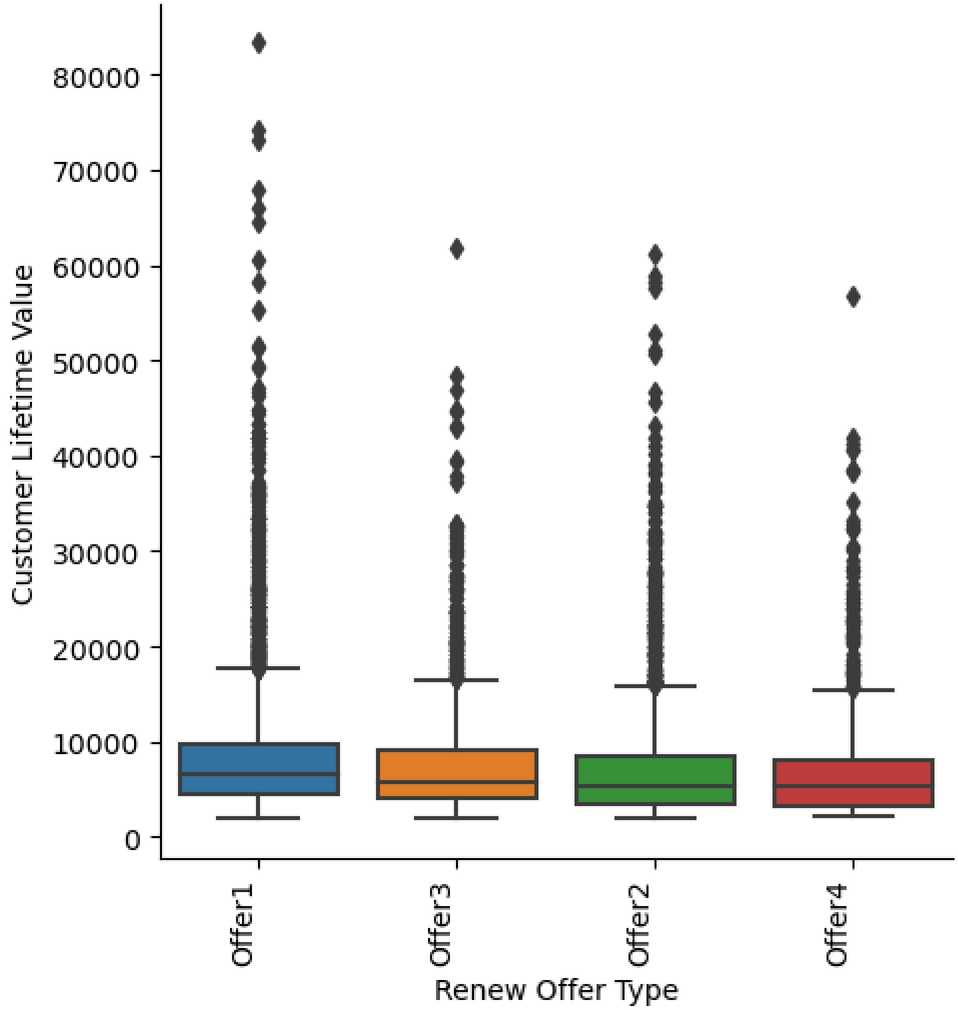
Out[31]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Corporate L3'),
  Text(1, 0, 'Personal L3'),
  Text(2, 0, 'Corporate L2'),
  Text(3, 0, 'Personal L1'),
  Text(4, 0, 'Special L2'),
  Text(5, 0, 'Corporate L1'),
  Text(6, 0, 'Personal L2'),
  Text(7, 0, 'Special L1'),
  Text(8, 0, 'Special L3')])
```

Loading [MathJax]/extensions/Safe.js



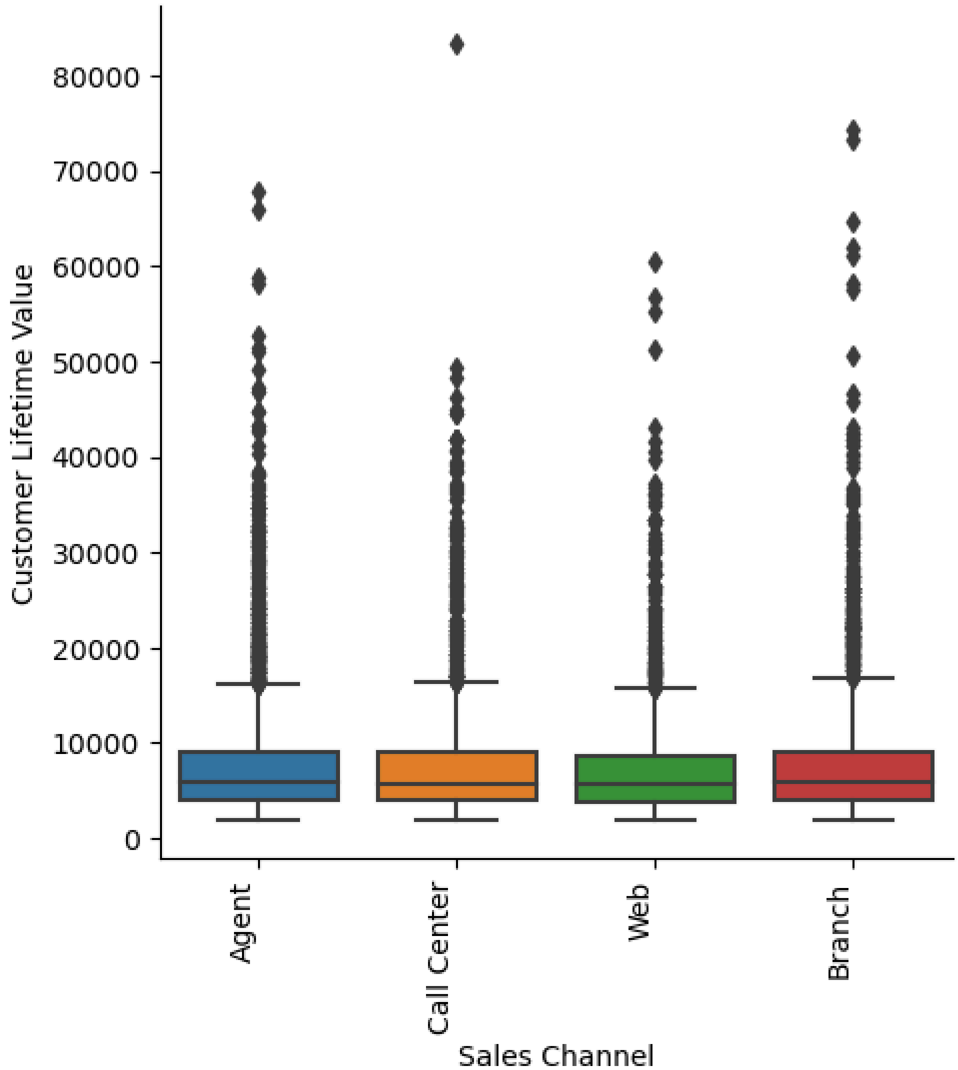
In [32]:
```
sns.catplot(data=data,
    x="Renew Offer Type", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

Out[32]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'Offer1'),
  Text(1, 0, 'Offer3'),
  Text(2, 0, 'Offer2'),
  Text(3, 0, 'Offer4')])
```



In [33]:
```
sns.catplot(data=data,
    x="Sales Channel", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

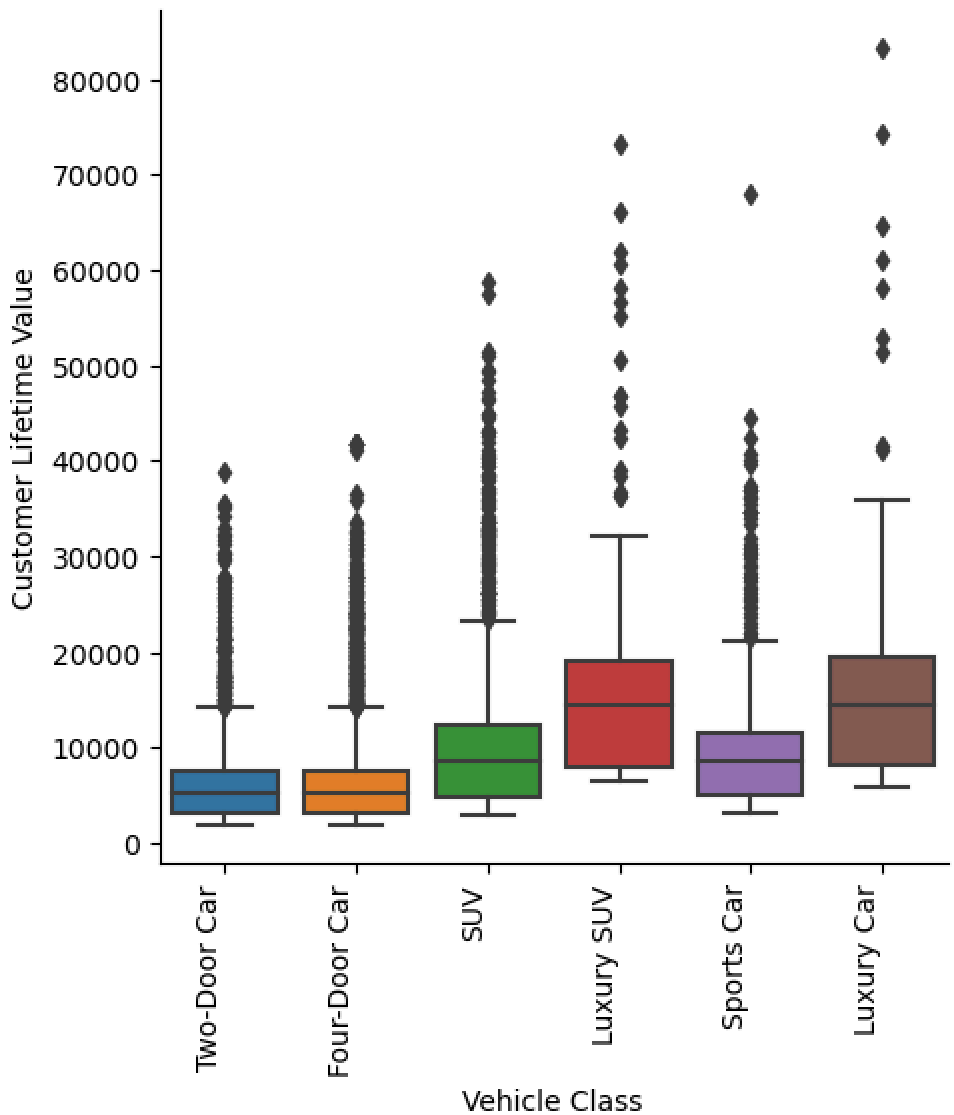
Out[33]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'Agent'),
  Text(1, 0, 'Call Center'),
  Text(2, 0, 'Web'),
  Text(3, 0, 'Branch')])
```



In [34]:
```
sns.catplot(data=data,
    x="Vehicle Class", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

Out[34]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Two-Door Car'),
  Text(1, 0, 'Four-Door Car'),
  Text(2, 0, 'SUV'),
  Text(3, 0, 'Luxury SUV'),
  Text(4, 0, 'Sports Car'),
  Text(5, 0, 'Luxury Car')])
```

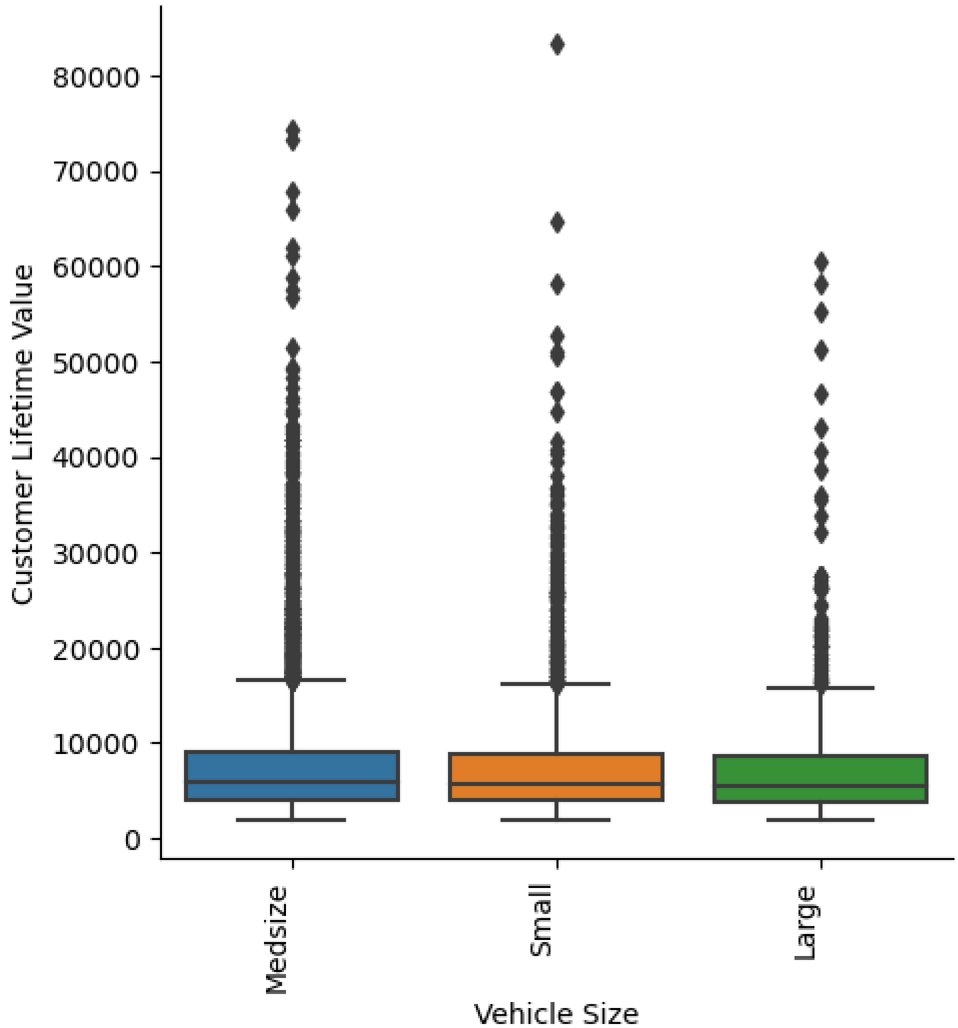
Loading [MathJax]/extensions/Safe.js



In [35]:
```
sns.catplot(data=data,
    x="Vehicle Size", y="Customer Lifetime Value", kind="box",
)
plt.xticks(rotation=90, ha="right")
```

Out[35]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'Medsize'), Text(1, 0, 'Small'), Text(2, 0, 'Large')])
```

Loading [MathJax]/extensions/Safe.js



In [36]:
```
data=data.drop(columns=["Customer","Effective To Date"],axis=1)
```

In [37]:
```
data.info()
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9134 entries, 0 to 9133
Data columns (total 22 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   State                          9134 non-null   object 
 1   Customer Lifetime Value        9134 non-null   float64
 2   Response                       9134 non-null   object 
 3   Coverage                       9134 non-null   object 
 4   Education                      9134 non-null   object 
 5   EmploymentStatus               9134 non-null   object 
 6   Gender                         9134 non-null   object 
 7   Income                         9134 non-null   int64  
 8   Location Code                  9134 non-null   object 
 9   Marital Status                 9134 non-null   object 
 10  Monthly Premium Auto           9134 non-null   int64  
 11  Months Since Last Claim        9134 non-null   int64  
 12  Months Since Policy Inception  9134 non-null   int64  
 13  Number of Open Complaints      9134 non-null   int64  
 14  Number of Policies             9134 non-null   int64  
 15  Policy Type                    9134 non-null   object 
 16  Policy                         9134 non-null   object 
 17  Renew Offer Type               9134 non-null   object 
 18  Sales Channel                  9134 non-null   object 
 19  Total Claim Amount             9134 non-null   float64
 20  Vehicle Class                  9134 non-null   object 
 21  Vehicle Size                   9134 non-null   object 
dtypes: float64(2), int64(6), object(14)
memory usage: 1.5+ MB
```

In [38]:
```
data2=pd.get_dummies(data,columns=["State","Response","Coverage","Education"
```

In [39]:
```
data2.info()
```

Loading [MathJax]/extensions/Safe.js

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9134 entries, 0 to 9133
Data columns (total 65 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Customer Lifetime Value         9134 non-null   float64
 1   Income                          9134 non-null   int64  
 2   Monthly Premium Auto            9134 non-null   int64  
 3   Months Since Last Claim         9134 non-null   int64  
 4   Months Since Policy Inception   9134 non-null   int64  
 5   Number of Open Complaints       9134 non-null   int64  
 6   Number of Policies              9134 non-null   int64  
 7   Total Claim Amount              9134 non-null   float64
 8   State_Arizona                   9134 non-null   uint8  
 9   State_California                9134 non-null   uint8  
 10  State_Nevada                    9134 non-null   uint8  
 11  State_Oregon                    9134 non-null   uint8  
 12  State_Washington                9134 non-null   uint8  
 13  Response_No                     9134 non-null   uint8  
 14  Response_Yes                    9134 non-null   uint8  
 15  Coverage_Basic                  9134 non-null   uint8  
 16  Coverage_Extended               9134 non-null   uint8  
 17  Coverage_Premium                9134 non-null   uint8  
 18  Education_Bachelor              9134 non-null   uint8  
 19  Education_College               9134 non-null   uint8  
 20  Education_Doctor                9134 non-null   uint8  
 21  Education_High School or Below  9134 non-null   uint8  
 22  Education_Master                9134 non-null   uint8  
 23  EmploymentStatus_Disabled       9134 non-null   uint8  
 24  EmploymentStatus_Employed       9134 non-null   uint8  
 25  EmploymentStatus_Medical Leave  9134 non-null   uint8  
 26  EmploymentStatus_Retired        9134 non-null   uint8  
 27  EmploymentStatus_Unemployed     9134 non-null   uint8  
 28  Gender_F                        9134 non-null   uint8  
 29  Gender_M                        9134 non-null   uint8  
 30  Location Code_Rural             9134 non-null   uint8  
 31  Location Code_Suburban          9134 non-null   uint8  
 32  Location Code_Urban             9134 non-null   uint8  
 33  Marital Status_Divorced         9134 non-null   uint8  
 34  Marital Status_Married          9134 non-null   uint8  
 35  Marital Status_Single           9134 non-null   uint8  
 36  Policy Type_Corporate Auto      9134 non-null   uint8  
 37  Policy Type_Personal Auto       9134 non-null   uint8  
 38  Policy Type_Special Auto        9134 non-null   uint8  
 39  Policy_Corporate L1             9134 non-null   uint8  
 40  Policy_Corporate L2             9134 non-null   uint8  
 41  Policy_Corporate L3             9134 non-null   uint8  
 42  Policy_Personal L1              9134 non-null   uint8  
 43  Policy_Personal L2              9134 non-null   uint8  
 44  Policy_Personal L3              9134 non-null   uint8  
 45  Policy_Special L1               9134 non-null   uint8  
 46  Policy_Special L2               9134 non-null   uint8  
 47  Policy_Special L3               9134 non-null   uint8  
 48  Renew Offer Type_Offer1         9134 non-null   uint8  
 49  Renew Offer Type_Offer2         9134 non-null   uint8  
          er Type_Offer3             9134 non-null   uint8  
```

```
 51  Renew Offer Type_Offer4         9134 non-null   uint8
 52  Sales Channel_Agent             9134 non-null   uint8
 53  Sales Channel_Branch            9134 non-null   uint8
 54  Sales Channel_Call Center       9134 non-null   uint8
 55  Sales Channel_Web               9134 non-null   uint8
 56  Vehicle Class_Four-Door Car     9134 non-null   uint8
 57  Vehicle Class_Luxury Car        9134 non-null   uint8
 58  Vehicle Class_Luxury SUV        9134 non-null   uint8
 59  Vehicle Class_SUV               9134 non-null   uint8
 60  Vehicle Class_Sports Car        9134 non-null   uint8
 61  Vehicle Class_Two-Door Car      9134 non-null   uint8
 62  Vehicle Size_Large              9134 non-null   uint8
 63  Vehicle Size_Medsize            9134 non-null   uint8
 64  Vehicle Size_Small              9134 non-null   uint8
dtypes: float64(2), int64(6), uint8(57)
memory usage: 1.1 MB
```

SPLIT DATA

In [40]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [41]:
```
# lets build our linear model
# independant variables
x = data2.drop(['Customer Lifetime Value'], axis=1)
# the dependent variable
y = data2[['Customer Lifetime Value']]
```

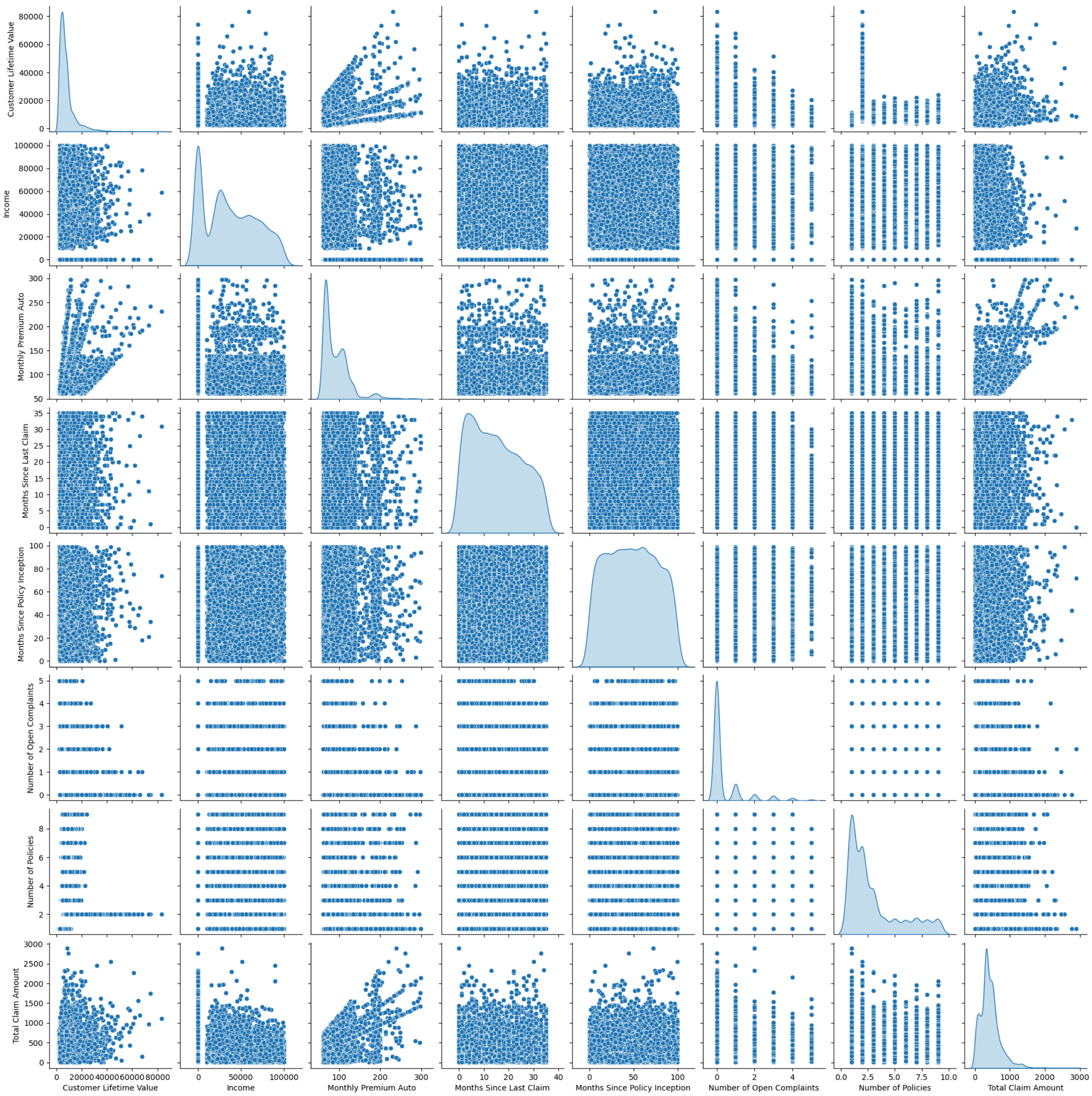
In [42]:
```
# Split x and y into training and test set in 70:30 ratio

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30,rar
```

In [43]:
```
sns.pairplot(data,diag_kind="kde")
```

Out[43]: <seaborn.axisgrid.PairGrid at 0x2b4232d0b80>

Loading [MathJax]/extensions/Safe.js



Mann-Whitney U test

It is a non parametric test of hypothesis testing. This test is used to investigate whether the two independent samples were selected from the population having the same distribution or not. The maximum values of U is n1*n2 and minimum value is Zero. It is also known as Mann Whitney Wilcoxon test. It is also known as Mann Whitney Wilcoxon Rank Sum test.

In [44]:
```
mannwhitneyu(data2['Response_No'], data2['Customer Lifetime Value'])
```

Out[44]:
```
MannwhitneyuResult(statistic=0.0, pvalue=0.0)
```

In [45]:
```
mannwhitneyu(data2['Response_Yes'], data2['Customer Lifetime Value'])
```

Out[45]:
```
MannwhitneyuResult(statistic=0.0, pvalue=0.0)
```

In [46]:
```
mannwhitneyu(data2['EmploymentStatus_Employed'], data2['Customer Lifetime Va
```

Loading [MathJax]/extensions/Safe.js

Out[46]: MannwhitneyuResult(statistic=0.0, pvalue=0.0)

In [47]:
```
mannwhitneyu(data2['Location Code_Rural'], data2['Customer Lifetime Value'])
```

Out[47]: MannwhitneyuResult(statistic=0.0, pvalue=0.0)

In [48]:
```
mannwhitneyu(data2['Policy Type_Corporate Auto'], data2['Customer Lifetime V
```

Out[48]: MannwhitneyuResult(statistic=0.0, pvalue=0.0)

In [49]:
```
mannwhitneyu(data2['Policy_Corporate L3'], data2['Customer Lifetime Value'])
```

Out[49]: MannwhitneyuResult(statistic=0.0, pvalue=0.0)

In [50]:
```
mannwhitneyu(data2['Policy_Special L3'], data2['Customer Lifetime Value'])
```

Out[50]: MannwhitneyuResult(statistic=0.0, pvalue=0.0)

In [51]:
```
pip install pingouin
```

```
Requirement already satisfied: pingouin in c:\users\hitesh sonar\anaconda3\li
b\site-packages (0.5.3)Note: you may need to restart the kernel to use update
d packages.

Requirement already satisfied: pandas-flavor>=0.2.0 in c:\users\hitesh sonar
\anaconda3\lib\site-packages (from pingouin) (0.6.0)
Requirement already satisfied: scikit-learn in c:\users\hitesh sonar\anaconda
3\lib\site-packages (from pingouin) (1.2.1)
Requirement already satisfied: scipy>=1.7 in c:\users\hitesh sonar\anaconda3
\lib\site-packages (from pingouin) (1.10.0)
Requirement already satisfied: tabulate in c:\users\hitesh sonar\anaconda3\li
b\site-packages (from pingouin) (0.8.10)
Requirement already satisfied: outdated in c:\users\hitesh sonar\anaconda3\li
b\site-packages (from pingouin) (0.2.2)
Requirement already satisfied: seaborn>=0.11 in c:\users\hitesh sonar\anacond
a3\lib\site-packages (from pingouin) (0.12.2)
Requirement already satisfied: numpy>=1.19 in c:\users\hitesh sonar\anaconda3
\lib\site-packages (from pingouin) (1.23.5)
Requirement already satisfied: pandas>=1.0 in c:\users\hitesh sonar\anaconda3
\lib\site-packages (from pingouin) (1.5.3)
Requirement already satisfied: matplotlib>=3.0.2 in c:\users\hitesh sonar\ana
conda3\lib\site-packages (from pingouin) (3.7.0)
Requirement already satisfied: statsmodels>=0.13 in c:\users\hitesh sonar\ana
conda3\lib\site-packages (from pingouin) (0.13.5)
Requirement already satisfied: cycler>=0.10 in c:\users\hitesh sonar\anaconda
3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\hitesh sonar\ana
conda3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (4.25.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hitesh sonar
\anaconda3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (2.8.2)
Requirement already satisfied: packaging>=20.0 in c:\users\hitesh sonar\anaco
nda3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (22.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\hitesh sonar\anac
onda3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hitesh sonar\anacond
a3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (9.4.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hitesh sonar\ana
conda3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (1.4.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\hitesh sonar\anac
onda3\lib\site-packages (from matplotlib>=3.0.2->pingouin) (1.0.5)
Requirement already satisfied: pytz>=2020.1 in c:\users\hitesh sonar\anaconda
3\lib\site-packages (from pandas>=1.0->pingouin) (2022.7)
Requirement already satisfied: xarray in c:\users\hitesh sonar\anaconda3\lib
\site-packages (from pandas-flavor>=0.2.0->pingouin) (2022.11.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\hitesh sonar\anaconda
3\lib\site-packages (from statsmodels>=0.13->pingouin) (0.5.3)
Requirement already satisfied: requests in c:\users\hitesh sonar\anaconda3\li
b\site-packages (from outdated->pingouin) (2.28.1)
Requirement already satisfied: setuptools>=44 in c:\users\hitesh sonar\anacon
da3\lib\site-packages (from outdated->pingouin) (65.6.3)
Requirement already satisfied: littleutils in c:\users\hitesh sonar\anaconda3
\lib\site-packages (from outdated->pingouin) (0.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\users\hitesh sonar\anacond
a3\lib\site-packages (from scikit-learn->pingouin) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hitesh sonar
\site-packages (from scikit-learn->pingouin) (2.2.0)
```

```
Requirement already satisfied: six in c:\users\hitesh sonar\anaconda3\lib\sit
e-packages (from patsy>=0.5.2->statsmodels>=0.13->pingouin) (1.16.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hitesh sonar\an
aconda3\lib\site-packages (from requests->outdated->pingouin) (2023.7.22)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\hitesh so
nar\anaconda3\lib\site-packages (from requests->outdated->pingouin) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\hitesh sonar
\anaconda3\lib\site-packages (from requests->outdated->pingouin) (1.26.14)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hitesh sonar\anaconda
3\lib\site-packages (from requests->outdated->pingouin) (3.4)
```

In [52]:

```
from pingouin import mwu
mwu(data2['Policy_Special L3'], data2['Customer Lifetime Value'])
```

Out[52]:

	U-val	alternative	p-val	RBC	CLES
MWU	0.0	two-sided	0.0	1.0	0.0

In [53]:

```
mwu(data2['Sales Channel_Agent'], data2['Customer Lifetime Value'])
```

Out[53]:

	U-val	alternative	p-val	RBC	CLES
MWU	0.0	two-sided	0.0	1.0	0.0

In [54]:

```
mwu(data2['Vehicle Size_Large'], data2['Customer Lifetime Value'])
```

Out[54]:

	U-val	alternative	p-val	RBC	CLES
MWU	0.0	two-sided	0.0	1.0	0.0

Kruskal Wallis test

It is a non parametric test of hypothesis testing. The test used for comparing is two or more samples of independent size. It is used for comparing more than 2 groups. One way anova is parametric equivalent of this test. Thatswhy it is also called one way anova on Rank. It uses rank instead of actual data. It doesnot assume to the population normally distributed. The test Statistic used here is H

In [55]:

```
mystate=data["State"]
myresponse=data["Response"]
mycoverage=data["Coverage"]
myeducation=data["Education"]
myemploymentstatus=data["EmploymentStatus"]
mygender=data["Gender"]
mylocationcode=data["Location Code"]
mymaritalstatus=data["Marital Status"]
mypolicytype=data["Policy Type"]
mypolicy=data["Policy"]
myrenewoffertype=data["Renew Offer Type"]
mysaleschannel=data["Sales Channel"]
myvehicleclass=data["Vehicle Class"]
myvehiclesize=data["Vehicle Size"]
```

In [56]:
```
myCrosstable=pd.crosstab(mystate,myresponse)
myCrosstable
```

Out[56]:

Response	No	Yes
State		
Arizona	1460	243
California	2694	456
Nevada	758	124
Oregon	2225	376
Washington	689	109

In [57]:
```
kruskal(mystate,myresponse)
```

Out[57]:
```
KruskalResult(statistic=1972.131590047541, pvalue=0.0)
```

In [58]:
```
myCrosstable2=pd.crosstab(myresponse,mycoverage)
myCrosstable2
```

Out[58]:

Coverage	Basic	Extended	Premium
Response			
No	4770	2352	704
Yes	798	390	120

In [59]:
```
kruskal(myresponse,mycoverage)
```

Out[59]:
```
KruskalResult(statistic=11011.731386365529, pvalue=0.0)
```

In [60]:
```
myCrosstable3=pd.crosstab(myemploymentstatus,myeducation)
myCrosstable3
```

Out[60]:

Education	Bachelor	College	Doctor	High School or Below	Master
EmploymentStatus					
Disabled	121	98	22	118	46
Employed	1702	1664	249	1528	555
Medical Leave	126	145	17	115	29
Retired	88	102	1	72	19
Unemployed	711	672	53	789	92

In [61]:
```
kruskal(myemploymentstatus,myeducation)
```

Out[61]:
```
KruskalResult(statistic=3642.2758854130657, pvalue=0.0)
```

Model Building

In [62]:
```
model_1 = LinearRegression()
model_1.fit(x_train, y_train)
```

Out[62]:
```
▼ LinearRegression
LinearRegression()
```

In [63]:
```
model_1.score(x_train, y_train)
```

Out[63]:
```
0.18022530782421198
```

In [64]:
```
model_1.score(x_test, y_test)
```

Out[64]:
```
0.13108006717705933
```

In [65]:
```
# fit to a simple linear model
```

In [66]:
```
for idx, col_name in enumerate(x_train.columns):
    print("The coefficient for {} is {}".format(col_name, model_1.coef_[0][i
```

```
The coefficient for Income is -0.002364226544782254
The coefficient for Monthly Premium Auto is 60.49919835815779
The coefficient for Months Since Last Claim is 5.078544972655536
The coefficient for Months Since Policy Inception is -1.3578794168717536
The coefficient for Number of Open Complaints is -245.91021869975927
The coefficient for Number of Policies is 46.321046738744755
The coefficient for Total Claim Amount is 0.17787822451239776
The coefficient for State_Arizona is -69.42522106821605
The coefficient for State_California is 105.88142407399592
The coefficient for State_Nevada is -148.95560719834685
The coefficient for State_Oregon is 27.544830566928795
The coefficient for State_Washington is 84.95457362565517
The coefficient for Response_No is 262.60899589730576
The coefficient for Response_Yes is -262.60899589737954
The coefficient for Coverage_Basic is -238.76137317521508
The coefficient for Coverage_Extended is 31.723772089839397
The coefficient for Coverage_Premium is 207.03760108532123
The coefficient for Education_Bachelor is 21.61415248035577
The coefficient for Education_College is -3.714789155607373
The coefficient for Education_Doctor is -229.4646952717393
The coefficient for Education_High School or Below is 397.11035547781745
The coefficient for Education_Master is -185.54502353064504
The coefficient for EmploymentStatus_Disabled is -331.6362639109152
The coefficient for EmploymentStatus_Employed is 353.2772577594029
The coefficient for EmploymentStatus_Medical Leave is 95.26807752219368
The coefficient for EmploymentStatus_Retired is 215.9785340135001
The coefficient for EmploymentStatus_Unemployed is -332.8876053851611
The coefficient for Gender_F is 50.31597075875152
The coefficient for Gender_M is -50.31597075870329
The coefficient for Location Code_Rural is 23.108445441369398
The coefficient for Location Code_Suburban is -277.1368318994438
The coefficient for Location Code_Urban is 254.02838645940085
The coefficient for Marital Status_Divorced is -23.144132182029907
The coefficient for Marital Status_Married is 156.25988008253273
The coefficient for Marital Status_Single is -133.11574790368
The coefficient for Policy Type_Corporate Auto is -161.02895518118495
The coefficient for Policy Type_Personal Auto is -104.83763826756875
The coefficient for Policy Type_Special Auto is 265.8665934488289
The coefficient for Policy_Corporate L1 is 537.2014553626551
The coefficient for Policy_Corporate L2 is -437.53425437812285
The coefficient for Policy_Corporate L3 is -260.69615616588965
The coefficient for Policy_Personal L1 is -106.91829976646997
The coefficient for Policy_Personal L2 is 29.815652560019004
The coefficient for Policy_Personal L3 is -27.734991060947802
The coefficient for Policy_Special L1 is -304.2757139213439
The coefficient for Policy_Special L2 is -358.15721490569337
The coefficient for Policy_Special L3 is 928.2995222759544
The coefficient for Renew Offer Type_Offer1 is 545.6407512739124
The coefficient for Renew Offer Type_Offer2 is -198.11053886881808
The coefficient for Renew Offer Type_Offer3 is 16.627751456692355
The coefficient for Renew Offer Type_Offer4 is -364.1579638607803
The coefficient for Sales Channel_Agent is -50.87452149926244
The coefficient for Sales Channel_Branch is 45.25348013369498
The coefficient for Sales Channel_Call Center is 127.08548653074723
The coefficient for Sales Channel_Web is -121.46444516498752
The coefficient for Vehicle Class_Four-Door Car is -1527.9114558556118
```

```
The coefficient for Vehicle Class_Luxury Car is 1596.570361940482
The coefficient for Vehicle Class_Luxury SUV is 1462.3373819574983
The coefficient for Vehicle Class_SUV is -182.59760995020997
The coefficient for Vehicle Class_Sports Car is -107.64055645002419
The coefficient for Vehicle Class_Two-Door Car is -1240.758121642512
The coefficient for Vehicle Size_Large is -217.69590795341787
The coefficient for Vehicle Size_Medsize is -0.8038027090869715
The coefficient for Vehicle Size_Small is 218.4997106627706
```

In [67]:
```
intercept = model_1.intercept_[0]

print("The intercept for our model is {}".format(intercept))
```

```
The intercept for our model is 3141.285454753741
```

In [68]:
```
mse=cross_val_score(model_1,x,y,scoring="neg_mean_squared_error",cv=5)
```

In [69]:
```
mean_mse=np.mean(mse)
print(mean_mse)
```

```
-39930860.0365188
```

In [70]:
```
y_pred=model_1.predict(x_test)
```

In [71]:
```
print(sqrt(mean_squared_error(y_test,y_pred)))
```

```
6157.898533877753
```

Ridge and Lasso Regression

Create a regularized RIDGE model and note the coefficients

In [72]:
```
ridge = Ridge(alpha=.3)
ridge.fit(x_train,y_train)
print ("Ridge model:", (ridge.coef_))
```

```
Ridge model: [[-2.36292782e-03  6.09076256e+01  5.07263209e+00 -1.35716617e+0
0
  -2.45945835e+02  4.63098335e+01  1.78168422e-01 -6.93056243e+01
   1.05827879e+02 -1.48833029e+02  2.74085916e+01  8.49021825e+01
   2.62510590e+02 -2.62510590e+02 -2.29338362e+02  3.27491937e+01
   1.96589169e+02  2.16630123e+01 -3.61621840e+00 -2.29621871e+02
   3.97095824e+02 -1.85520748e+02 -3.31298625e+02  3.53239828e+02
   9.49397601e+01  2.15809717e+02 -3.32690680e+02  5.02173500e+01
  -5.02173500e+01  2.31904207e+01 -2.77131000e+02  2.53940579e+02
  -2.33045128e+01  1.56315380e+02 -1.33010867e+02 -1.61153229e+02
  -1.04860009e+02  2.66013238e+02  5.36401892e+02 -4.37089078e+02
  -2.60466044e+02 -1.06949185e+02  2.98825040e+01 -2.77933277e+01
  -3.02263245e+02 -3.56862429e+02  9.25138913e+02  5.45528127e+02
  -1.98169059e+02  1.67921059e+01 -3.64151174e+02 -5.08320267e+01
   4.52603887e+01  1.26966663e+02 -1.21395025e+02 -1.50260302e+03
   1.56361486e+03  1.43023094e+03 -1.75202748e+02 -1.00659959e+02
  -1.21538008e+03 -2.17511464e+02 -8.52084466e-01  2.18363548e+02]]
```

Create a regularized LASSO model and note the coefficients

In [73]:

```
lasso = Lasso(alpha=0.1)
lasso.fit(x_train,y_train)
print ("Lasso model:", (lasso.coef_))

# Observe, many of the coefficients have become 0 indicating drop of those c
```

```
Lasso model: [-2.34964001e-03  6.11927627e+01  5.06460504e+00 -1.35491569e+00
 -2.45839951e+02  4.62801540e+01  1.77417414e-01 -9.59612887e+01
  7.82352970e+01 -1.75150038e+02  0.00000000e+00  5.64756990e+01
  5.24082884e+02 -0.00000000e+00 -2.56165124e+02 -0.00000000e+00
  1.54974873e+02  2.48600890e+01 -0.00000000e+00 -2.24238451e+02
  4.00270409e+02 -1.80836913e+02 -4.24011014e+02  2.57891791e+02
  0.00000000e+00  1.17654913e+02 -4.26912454e+02  9.98879323e+01
 -1.74818665e-13 -0.00000000e+00 -2.99656952e+02  2.30357834e+02
  0.00000000e+00  1.79581884e+02 -1.09134880e+02 -2.12437493e+02
  0.00000000e+00  8.18817350e+01  7.18362527e+02 -2.51883755e+02
 -7.57877734e+01 -7.84994177e+01  5.73235584e+01  0.00000000e+00
  0.00000000e+00 -3.45199546e+01  1.23845772e+03  7.19010989e+02
 -2.42342634e+01  1.90027228e+02 -1.89538958e+02 -6.70597675e+01
  2.83547204e+01  1.09653684e+02 -1.37196719e+02 -1.31569140e+03
  1.71183793e+03  1.57806913e+03 -0.00000000e+00  7.14450128e+01
 -1.02842660e+03 -2.15740439e+02 -0.00000000e+00  2.18447210e+02]
```

Let us compare their scores

In [74]:

```
print(model_1.score(x_train, y_train))
print(model_1.score(x_test, y_test))
```

```
0.18022530782421198
0.13108006717705933
```

In [75]:
```
print(ridge.score(x_train, y_train))
print(ridge.score(x_test, y_test))
```

```
0.18022513680888286
0.13112164609618493
```

In [76]:
```
print(lasso.score(x_train, y_train))
print(lasso.score(x_test, y_test))
```

```
0.18022471765577797
0.13114906070537347
```

Polynomial Regression

In [77]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model

poly = PolynomialFeatures(degree=2, interaction_only=True)
x_train2 = poly.fit_transform(x_train)
x_test2 = poly.fit_transform(x_test)

poly_model2 = linear_model.LinearRegression()

poly_model2.fit(x_train2, y_train)

#In sample (training) R^2 will always improve with the number of variables!
print(poly_model2.score(x_train2, y_train))
```

```
0.37427978292484587
```

In [78]:
```
#Out off sample (testing) R^2 is our measure of sucess and does improve
print(poly_model2.score(x_test2, y_test))
```

```
-0.1853173793400993
```

poly3 = PolynomialFeatures(degree=3, interaction_only=True) x_train3 = poly3.fit_transform(x_train) x_test3 = poly3.fit_transform(x_test) poly_model3 = linear_model.LinearRegression() poly_model3.fit(x_train3, y_train) #In sample (training) R^2 will always improve with the number of variables! print(poly_model3.score(x_train3, y_train))

Support Vector Regressor

In [79]:
```
from sklearn.svm import SVR
```

In [80]:
```
svr=SVR()
svr.fit(x_train, y_train)
svr.score(x_test,y_test)
```

Out[80]: -0.10039101301165809

In [81]:
```
print("Support Vector Regressor Accuracy : {:.2f}%".format(svr.score(x_test,
```

```
Support Vector Regressor Accuracy : -10.04%
```

Loading [MathJax]/extensions/Safe.js

In [82]:
```
svr1=SVR(
    kernel='rbf',
    degree=3,
    gamma='auto',
    coef0=0.0,
    tol=0.001,
    C=1.0,
    epsilon=0.1,
    shrinking=True,
    cache_size=200,
    verbose=False,
    max_iter=-1,
)
```

In [83]:
```
svr1.fit(x_train, y_train)
svr1.score(x_test,y_test)
```

Out[83]: -0.0994232744813397

In [84]:
```
svr1.fit(x_train, y_train)
print("SVR Accuracy with Auto : {:.2f}%".format(svr1.score(x_test,y_test)*10
```

SVR Accuracy with Auto : -9.94%

In [85]:
```
y_pred1=svr.predict(x_test)
```

In [86]:
```
print(sqrt(mean_squared_error(y_test,y_pred1)))
```

6929.727643197682

In [87]:
```
# {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}
```

In [88]:
```
svr2=SVR(
    kernel='linear',
    degree=3,
    gamma='scale',
    coef0=0.0,
    tol=0.001,
    C=1.0,
    epsilon=0.1,
    shrinking=True,
    cache_size=200,
    verbose=False,
    max_iter=-1,
)
```

In [89]:
```
# svr2.fit(x_train, y_train)
# svr2.score(x_test,y_test)
```

In [90]:
```
# svr2.fit(x_train, y_train)
# print("SVR Accuracy with Linear Kernel : {:.2f}%".format(svr2.score(x_test
```

Loading [MathJax]/extensions/Safe.js

Decision Tree Regressor

In [91]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [92]:
```
dtr=DecisionTreeRegressor(max_depth=6)
```

In [93]:
```
dtr.fit(x_train, y_train)
```

Out[93]:
```
▼ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=6)
```

In [94]:
```
print("Decision Tree Regressor : {:.2f}%".format(dtr.score(x_test,y_test)*10
```

```
Decision Tree Regressor : 66.50%
```

In [95]:
```
y_pred2=dtr.predict(x_test)
```

In [96]:
```
print(sqrt(mean_squared_error(y_test,y_pred2)))
```

```
3823.53363245901
```

In [97]:
```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(dtr,filled=True)
```

Out[97]:

```
[Text(0.5, 0.9285714285714286, 'x[5] <= 1.5\nsquared_error = 48727636.976\n
samples = 6393\nvalue = 8046.92'),
 Text(0.25, 0.7857142857142857, 'x[1] <= 124.5\nsquared_error = 1974859.521
\nsamples = 2263\nvalue = 3580.027'),
 Text(0.125, 0.6428571428571429, 'x[1] <= 87.5\nsquared_error = 532494.138
\nsamples = 1966\nvalue = 3163.816'),
 Text(0.0625, 0.5, 'x[1] <= 72.5\nsquared_error = 91206.936\nsamples = 1254
\nvalue = 2684.768'),
 Text(0.03125, 0.35714285714285715, 'x[1] <= 67.5\nsquared_error = 38358.10
7\nsamples = 872\nvalue = 2542.577'),
 Text(0.015625, 0.21428571428571427, 'x[23] <= 0.5\nsquared_error = 24978.5
21\nsamples = 530\nvalue = 2446.996'),
 Text(0.0078125, 0.07142857142857142, 'squared_error = 17996.614\nsamples =
199\nvalue = 2296.35'),
 Text(0.0234375, 0.07142857142857142, 'squared_error = 7329.371\nsamples =
331\nvalue = 2537.566'),
 Text(0.046875, 0.21428571428571427, 'x[23] <= 0.5\nsquared_error = 22994.3
38\nsamples = 342\nvalue = 2690.7'),
 Text(0.0390625, 0.07142857142857142, 'squared_error = 18377.032\nsamples =
124\nvalue = 2540.87'),
 Text(0.0546875, 0.07142857142857142, 'squared_error = 5588.211\nsamples =
218\nvalue = 2775.925'),
 Text(0.09375, 0.35714285714285715, 'x[1] <= 78.5\nsquared_error = 60340.02
7\nsamples = 382\nvalue = 3009.35'),
 Text(0.078125, 0.21428571428571427, 'x[0] <= 5156.0\nsquared_error = 2887
5.548\nsamples = 192\nvalue = 2855.365'),
 Text(0.0703125, 0.07142857142857142, 'squared_error = 24833.157\nsamples =
47\nvalue = 2635.77'),
 Text(0.0859375, 0.07142857142857142, 'squared_error = 9488.826\nsamples =
145\nvalue = 2926.544'),
 Text(0.109375, 0.21428571428571427, 'x[0] <= 6381.0\nsquared_error = 4396
1.395\nsamples = 190\nvalue = 3164.956'),
 Text(0.1015625, 0.07142857142857142, 'squared_error = 23239.576\nsamples =
50\nvalue = 2896.238'),
 Text(0.1171875, 0.07142857142857142, 'squared_error = 16362.51\nsamples =
140\nvalue = 3260.927'),
 Text(0.1875, 0.5, 'x[1] <= 105.5\nsquared_error = 193665.932\nsamples = 71
2\nvalue = 4007.533'),
 Text(0.15625, 0.35714285714285715, 'x[0] <= 16936.5\nsquared_error = 7862
1.967\nsamples = 378\nvalue = 3701.224'),
 Text(0.140625, 0.21428571428571427, 'x[1] <= 94.5\nsquared_error = 62501.8
15\nsamples = 114\nvalue = 3468.946'),
 Text(0.1328125, 0.07142857142857142, 'squared_error = 32404.997\nsamples =
37\nvalue = 3260.997'),
 Text(0.1484375, 0.07142857142857142, 'squared_error = 46200.045\nsamples =
77\nvalue = 3568.87'),
 Text(0.171875, 0.21428571428571427, 'x[1] <= 98.5\nsquared_error = 52224.6
61\nsamples = 264\nvalue = 3801.526'),
 Text(0.1640625, 0.07142857142857142, 'squared_error = 23345.736\nsamples =
153\nvalue = 3652.234'),
 Text(0.1796875, 0.07142857142857142, 'squared_error = 18964.222\nsamples =
111\nvalue = 4007.306'),
 Text(0.21875, 0.35714285714285715, 'x[0] <= 11716.0\nsquared_error = 9750
6.043\nsamples = 334\nvalue = 4354.194'),
 Text(0.203125, 0.21428571428571427, 'x[1] <= 113.5\nsquared_error = 91155.
= 83\nvalue = 3990.861'),
```

```
 Text(0.1953125, 0.07142857142857142, 'squared_error = 47669.993\nsamples =
43\nvalue = 3803.917'),
 Text(0.2109375, 0.07142857142857142, 'squared_error = 59945.541\nsamples =
40\nvalue = 4191.827'),
 Text(0.234375, 0.21428571428571427, 'x[1] <= 113.5\nsquared_error = 41518.
036\nsamples = 251\nvalue = 4474.34'),
 Text(0.2265625, 0.07142857142857142, 'squared_error = 15632.96\nsamples =
134\nvalue = 4329.657'),
 Text(0.2421875, 0.07142857142857142, 'squared_error = 19731.14\nsamples =
117\nvalue = 4640.046'),
 Text(0.375, 0.6428571428571429, 'x[1] <= 169.0\nsquared_error = 2785234.71
4\nsamples = 297\nvalue = 6335.148'),
 Text(0.3125, 0.5, 'x[1] <= 145.0\nsquared_error = 240850.491\nsamples = 18
0\nvalue = 5235.195'),
 Text(0.28125, 0.35714285714285715, 'x[23] <= 0.5\nsquared_error = 104459.5
31\nsamples = 149\nvalue = 5074.129'),
 Text(0.265625, 0.21428571428571427, 'x[1] <= 131.5\nsquared_error = 58206.
053\nsamples = 61\nvalue = 4796.022'),
 Text(0.2578125, 0.07142857142857142, 'squared_error = 54491.599\nsamples =
24\nvalue = 4621.452'),
 Text(0.2734375, 0.07142857142857142, 'squared_error = 28026.156\nsamples =
37\nvalue = 4909.256'),
 Text(0.296875, 0.21428571428571427, 'x[1] <= 134.5\nsquared_error = 45744.
378\nsamples = 88\nvalue = 5266.909'),
 Text(0.2890625, 0.07142857142857142, 'squared_error = 16516.578\nsamples =
48\nvalue = 5106.564'),
 Text(0.3046875, 0.07142857142857142, 'squared_error = 12942.607\nsamples =
40\nvalue = 5459.322'),
 Text(0.34375, 0.35714285714285715, 'x[0] <= 17198.5\nsquared_error = 17240
2.875\nsamples = 31\nvalue = 6009.349'),
 Text(0.328125, 0.21428571428571427, 'x[18] <= 0.5\nsquared_error = 68109.0
68\nsamples = 10\nvalue = 5566.221'),
 Text(0.3203125, 0.07142857142857142, 'squared_error = 24252.703\nsamples =
7\nvalue = 5713.833'),
 Text(0.3359375, 0.07142857142857142, 'squared_error = 966.641\nsamples = 3
\nvalue = 5221.791'),
 Text(0.359375, 0.21428571428571427, 'x[1] <= 162.5\nsquared_error = 84033.
825\nsamples = 21\nvalue = 6220.363'),
 Text(0.3515625, 0.07142857142857142, 'squared_error = 21599.417\nsamples =
14\nvalue = 6058.078'),
 Text(0.3671875, 0.07142857142857142, 'squared_error = 50883.834\nsamples =
7\nvalue = 6544.933'),
 Text(0.4375, 0.5, 'x[1] <= 227.5\nsquared_error = 1974629.623\nsamples = 1
17\nvalue = 8027.384'),
 Text(0.40625, 0.35714285714285715, 'x[1] <= 208.0\nsquared_error = 376315.
371\nsamples = 91\nvalue = 7372.907'),
 Text(0.390625, 0.21428571428571427, 'x[0] <= 22365.5\nsquared_error = 2225
11.0\nsamples = 78\nvalue = 7205.089'),
 Text(0.3828125, 0.07142857142857142, 'squared_error = 169003.107\nsamples
= 21\nvalue = 6686.609'),
 Text(0.3984375, 0.07142857142857142, 'squared_error = 106696.409\nsamples
= 57\nvalue = 7396.108'),
 Text(0.421875, 0.21428571428571427, 'x[1] <= 212.5\nsquared_error = 11630
6.174\nsamples = 13\nvalue = 8379.812'),
 Text(0.4140625, 0.07142857142857142, 'squared_error = 0.0\nsamples = 3\nva
6'),
```

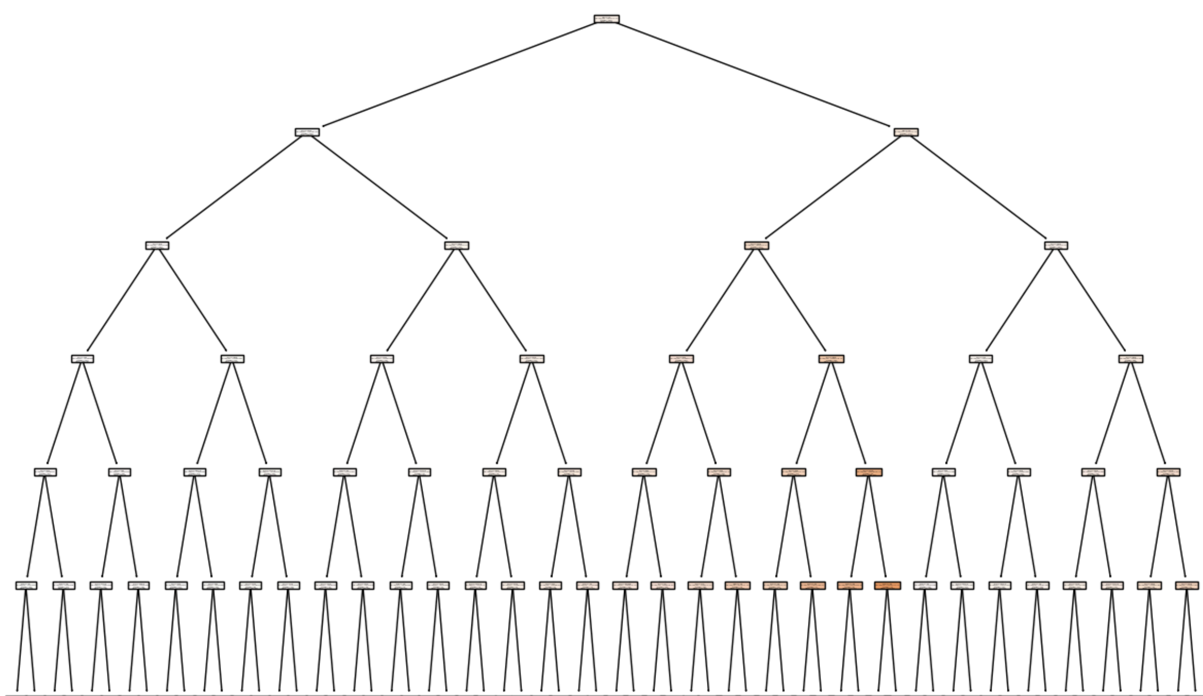
```
 Text(0.4296875, 0.07142857142857142, 'squared_error = 37622.82\nsamples = 
10\nvalue = 8541.706'),
 Text(0.46875, 0.35714285714285715, 'x[1] <= 275.5\nsquared_error = 822364.
802\nsamples = 26\nvalue = 10318.055'),
 Text(0.453125, 0.21428571428571427, 'x[3] <= 42.0\nsquared_error = 389388.
649\nsamples = 14\nvalue = 9613.166'),
 Text(0.4453125, 0.07142857142857142, 'squared_error = 117530.427\nsamples 
= 5\nvalue = 10266.112'),
 Text(0.4609375, 0.07142857142857142, 'squared_error = 171980.373\nsamples 
= 9\nvalue = 9250.419'),
 Text(0.484375, 0.21428571428571427, 'x[13] <= 0.5\nsquared_error = 71531.8
88\nsamples = 12\nvalue = 11140.425'),
 Text(0.4765625, 0.07142857142857142, 'squared_error = 9656.16\nsamples = 3
\nvalue = 11594.539'),
 Text(0.4921875, 0.07142857142857142, 'squared_error = 503.858\nsamples = 9
\nvalue = 10989.053'),
 Text(0.75, 0.7857142857142857, 'x[5] <= 2.5\nsquared_error = 57421553.956
\nsamples = 4130\nvalue = 10494.517'),
 Text(0.625, 0.6428571428571429, 'x[1] <= 102.5\nsquared_error = 89924250.0
72\nsamples = 1626\nvalue = 15764.183'),
 Text(0.5625, 0.5, 'x[1] <= 81.5\nsquared_error = 37968454.523\nsamples = 1
094\nvalue = 12722.563'),
 Text(0.53125, 0.35714285714285715, 'x[1] <= 69.5\nsquared_error = 3089323
0.242\nsamples = 788\nvalue = 11718.551'),
 Text(0.515625, 0.21428571428571427, 'x[6] <= 681.668\nsquared_error = 2833
6818.596\nsamples = 489\nvalue = 11125.849'),
 Text(0.5078125, 0.07142857142857142, 'squared_error = 28343997.911\nsample
s = 480\nvalue = 11034.438'),
 Text(0.5234375, 0.07142857142857142, 'squared_error = 3740063.355\nsamples
= 9\nvalue = 16001.113'),
 Text(0.546875, 0.21428571428571427, 'x[4] <= 2.5\nsquared_error = 3355997
8.816\nsamples = 299\nvalue = 12687.887'),
 Text(0.5390625, 0.07142857142857142, 'squared_error = 34592789.413\nsample
s = 276\nvalue = 13041.478'),
 Text(0.5546875, 0.07142857142857142, 'squared_error = 1661998.523\nsamples
= 23\nvalue = 8444.783'),
 Text(0.59375, 0.35714285714285715, 'x[2] <= 33.5\nsquared_error = 4690766
3.91\nsamples = 306\nvalue = 15308.057'),
 Text(0.578125, 0.21428571428571427, 'x[6] <= 4.41\nsquared_error = 4283522
4.969\nsamples = 290\nvalue = 14972.809'),
 Text(0.5703125, 0.07142857142857142, 'squared_error = 0.0\nsamples = 1\nva
lue = 35186.256'),
 Text(0.5859375, 0.07142857142857142, 'squared_error = 41564768.176\nsample
s = 289\nvalue = 14902.866'),
 Text(0.609375, 0.21428571428571427, 'x[55] <= 0.5\nsquared_error = 8176111
1.214\nsamples = 16\nvalue = 21384.439'),
 Text(0.6015625, 0.07142857142857142, 'squared_error = 43932760.099\nsample
s = 9\nvalue = 27550.134'),
 Text(0.6171875, 0.07142857142857142, 'squared_error = 18677669.531\nsample
s = 7\nvalue = 13457.116'),
 Text(0.6875, 0.5, 'x[1] <= 160.5\nsquared_error = 138619033.751\nsamples =
532\nvalue = 22018.944'),
 Text(0.65625, 0.35714285714285715, 'x[2] <= 33.5\nsquared_error = 8661080
9.734\nsamples = 449\nvalue = 19829.704'),
 Text(0.640625, 0.21428571428571427, 'x[12] <= 0.5\nsquared_error = 7930036
[illegible]es = 432\nvalue = 19461.54'),
```

Loading [MathJax]/extensions/Safe.js

```
 Text(0.6328125, 0.07142857142857142, 'squared_error = 38579359.728\nsample
s = 69\nvalue = 15954.002'),
 Text(0.6484375, 0.07142857142857142, 'squared_error = 84257648.135\nsample
s = 363\nvalue = 20128.262'),
 Text(0.671875, 0.21428571428571427, 'x[14] <= 0.5\nsquared_error = 1814085
55.612\nsamples = 17\nvalue = 29185.406'),
 Text(0.6640625, 0.07142857142857142, 'squared_error = 162750899.514\nsampl
es = 9\nvalue = 36440.796'),
 Text(0.6796875, 0.07142857142857142, 'squared_error = 76554304.026\nsample
s = 8\nvalue = 21023.093'),
 Text(0.71875, 0.35714285714285715, 'x[1] <= 215.5\nsquared_error = 2537809
22.204\nsamples = 83\nvalue = 33861.944'),
 Text(0.703125, 0.21428571428571427, 'x[20] <= 0.5\nsquared_error = 2076705
76.953\nsamples = 61\nvalue = 31033.325'),
 Text(0.6953125, 0.07142857142857142, 'squared_error = 237095140.924\nsampl
es = 45\nvalue = 33776.317'),
 Text(0.7109375, 0.07142857142857142, 'squared_error = 44236712.624\nsample
s = 16\nvalue = 23318.662'),
 Text(0.734375, 0.21428571428571427, 'x[20] <= 0.5\nsquared_error = 2979350
71.223\nsamples = 22\nvalue = 41704.931'),
 Text(0.7265625, 0.07142857142857142, 'squared_error = 140232651.642\nsampl
es = 14\nvalue = 34903.962'),
 Text(0.7421875, 0.07142857142857142, 'squared_error = 351320903.096\nsampl
es = 8\nvalue = 53606.626'),
 Text(0.875, 0.6428571428571429, 'x[1] <= 100.5\nsquared_error = 6573708.70
6\nsamples = 2504\nvalue = 7072.601'),
 Text(0.8125, 0.5, 'x[1] <= 77.5\nsquared_error = 782173.697\nsamples = 164
9\nvalue = 5671.545'),
 Text(0.78125, 0.35714285714285715, 'x[1] <= 69.5\nsquared_error = 213434.2
83\nsamples = 1145\nvalue = 5197.868'),
 Text(0.765625, 0.21428571428571427, 'x[23] <= 0.5\nsquared_error = 131518.
209\nsamples = 700\nvalue = 4974.696'),
 Text(0.7578125, 0.07142857142857142, 'squared_error = 99213.555\nsamples =
252\nvalue = 4642.727'),
 Text(0.7734375, 0.07142857142857142, 'squared_error = 52831.585\nsamples =
448\nvalue = 5161.428'),
 Text(0.796875, 0.21428571428571427, 'x[0] <= 15127.0\nsquared_error = 1407
02.394\nsamples = 445\nvalue = 5548.927'),
 Text(0.7890625, 0.07142857142857142, 'squared_error = 112577.665\nsamples
= 134\nvalue = 5127.029'),
 Text(0.8046875, 0.07142857142857142, 'squared_error = 43081.93\nsamples =
311\nvalue = 5730.709'),
 Text(0.84375, 0.35714285714285715, 'x[1] <= 87.5\nsquared_error = 406508.9
89\nsamples = 504\nvalue = 6747.655'),
 Text(0.828125, 0.21428571428571427, 'x[0] <= 13595.0\nsquared_error = 1621
23.572\nsamples = 251\nvalue = 6318.427'),
 Text(0.8203125, 0.07142857142857142, 'squared_error = 139092.143\nsamples
= 64\nvalue = 5856.837'),
 Text(0.8359375, 0.07142857142857142, 'squared_error = 72128.316\nsamples =
187\nvalue = 6476.404'),
 Text(0.859375, 0.21428571428571427, 'x[23] <= 0.5\nsquared_error = 284846.
559\nsamples = 253\nvalue = 7173.49'),
 Text(0.8515625, 0.07142857142857142, 'squared_error = 195244.773\nsamples
= 93\nvalue = 6657.846'),
 Text(0.8671875, 0.07142857142857142, 'squared_error = 92548.652\nsamples =
7473.208'),
```

Loading [MathJax]/extensions/Safe.js

```
 Text(0.9375, 0.5, 'x[1] <= 155.5\nsquared_error = 6656068.632\nsamples = 8
55\nvalue = 9774.756'),
 Text(0.90625, 0.35714285714285715, 'x[1] <= 122.5\nsquared_error = 867047.
829\nsamples = 725\nvalue = 8811.681'),
 Text(0.890625, 0.21428571428571427, 'x[1] <= 112.5\nsquared_error = 37757
7.964\nsamples = 573\nvalue = 8462.172'),
 Text(0.8828125, 0.07142857142857142, 'squared_error = 250850.905\nsamples
= 346\nvalue = 8161.201'),
 Text(0.8984375, 0.07142857142857142, 'squared_error = 222218.238\nsamples
= 227\nvalue = 8920.921'),
 Text(0.921875, 0.21428571428571427, 'x[0] <= 17120.5\nsquared_error = 5157
63.742\nsamples = 152\nvalue = 10129.239'),
 Text(0.9140625, 0.07142857142857142, 'squared_error = 347712.309\nsamples
= 53\nvalue = 9478.533'),
 Text(0.9296875, 0.07142857142857142, 'squared_error = 257699.118\nsamples
= 99\nvalue = 10477.596'),
 Text(0.96875, 0.35714285714285715, 'x[1] <= 211.5\nsquared_error = 492071
6.349\nsamples = 130\nvalue = 15145.753'),
 Text(0.953125, 0.21428571428571427, 'x[1] <= 186.5\nsquared_error = 146991
5.596\nsamples = 99\nvalue = 14203.034'),
 Text(0.9453125, 0.07142857142857142, 'squared_error = 891314.58\nsamples =
43\nvalue = 13329.634'),
 Text(0.9609375, 0.07142857142857142, 'squared_error = 878687.174\nsamples
= 56\nvalue = 14873.682'),
 Text(0.984375, 0.21428571428571427, 'x[1] <= 264.5\nsquared_error = 403904
0.093\nsamples = 31\nvalue = 18156.369'),
 Text(0.9765625, 0.07142857142857142, 'squared_error = 2022282.841\nsamples
= 28\nvalue = 17689.297'),
 Text(0.9921875, 0.07142857142857142, 'squared_error = 1822102.395\nsamples
= 3\nvalue = 22515.713')]
```



Random Forest Regressor

In [98]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [99]:
```
rf=RandomForestRegressor(n_estimators=1000,random_state=1,max_depth=6)
```

In [100…
```
rf.fit(x_train, y_train)
print("Random Forest Classifier : {:.2f}%".format(rf.score(x_test,y_test)*10
```

```
Random Forest Classifier : 69.10%
```

In [101…
```
y_pred3=rf.predict(x_test)
```

In [102…
```
print(sqrt(mean_squared_error(y_test,y_pred3)))
```

```
3672.021549849275
```

Bagging

In [103…
```
from sklearn.ensemble import BaggingRegressor
```

In [104…
```
bgr=BaggingRegressor(n_estimators=23)
```

In [105…
```
bgr.fit(x_train, y_train)
```

Out[105]:
```
▼ BaggingRegressor
BaggingRegressor(n_estimators=23)
```

In [106…
```
print(bgr.score(x_train,y_train))
print(bgr.score(x_test,y_test))
```

```
0.9492370914268227
0.6980046307054949
```

In [107…
```
print("Bagging Regressor : {:.2f}%".format(bgr.score(x_test,y_test)*100))
```

```
Bagging Regressor : 69.80%
```

Note: This model is overfit

In [108…
```
y_pred4=bgr.predict(x_test)
```

In [109…
```
print(sqrt(mean_squared_error(y_test,y_pred4)))
```

```
3630.3022969372946
```

XG-Boost

In [110…
```
import xgboost
```

In [111…
```
xgbr=xgboost.XGBRFRegressor()
```

In [112…
```
xgbr.fit(x_train, y_train)
```

Out[112]:
```
▼ XGBRFRegressor

XGBRFRegressor(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bytree=None, devic
e=None,
               early_stopping_rounds=None, enable_categorical=Fals
e,
               eval_metric=None, feature_types=None, gamma=None,
               grow_policy=None, importance_type=None,
               interaction_constraints=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
```

In [113…
```
print(xgbr.score(x_train,y_train))
print(xgbr.score(x_test,y_test))
```

```
0.7401052820669541
0.687614200416732
```

In [114…
```
print("Boosting Regressor : {:.2f}%".format(xgbr.score(x_test,y_test)*100))
```

```
Boosting Regressor : 68.76%
```

Note: This model is suitable for this dataset

In [115…
```
y_pred5=xgbr.predict(x_test)
```

In [116…
```
print(sqrt(mean_squared_error(y_test,y_pred5)))
```

```
3692.2261204764727
```

Adaboost

In [117…
```
from sklearn.ensemble import AdaBoostRegressor
```

In [118…
```
abr=AdaBoostRegressor()
```

In [119…
```
abr.fit(x_train, y_train)
```

Out[119]:
```
▼ AdaBoostRegressor

AdaBoostRegressor()
```

In [120…
```
print(abr.score(x_train, y_train))
print(abr.score(x_test,y_test))
```

```
0.5400681441468764
0.5019552089214354
```

In [121…
```
print("Adaboost Regressor : {:.2f}%".format(abr.score(x_test,y_test)*100))
```

```
Adaboost Regressor : 50.20%
```

In [122…
```
y_pred6=abr.predict(x_test)
```

In [123…
```
print(sqrt(mean_squared_error(y_test,y_pred6)))
```

```
4662.049171043653
```

Gradient Boosting Regressor

In [124…
```
from sklearn.ensemble import GradientBoostingRegressor
```

In [125…
```
gbr=GradientBoostingRegressor()
```

In [126…
```
gbr.fit(x_train, y_train)
```

Out[126]:
```
▼ GradientBoostingRegressor
GradientBoostingRegressor()
```

In [127…
```
print(gbr.score(x_train, y_train))
print(gbr.score(x_test,y_test))
```

```
0.7388896218601886
0.6824614451074824
```

In [128…
```
print("Gradient boost Regressor : {:.2f}%".format(gbr.score(x_test,y_test)*1
```

```
Gradient boost Regressor : 68.25%
```

In [129…
```
y_pred7=gbr.predict(x_test)
```

In [130…
```
print(sqrt(mean_squared_error(y_test,y_pred7)))
```

```
3722.5529215774272
```

Note:- Only Gradient Boost Regressor Model will be suitable for this data

*********______________************________________**************

In [131…
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [132…
```
from sklearn.model_selection import cross_val_score
```

Loading [MathJax]/extensions/Safe.js

In [133…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9134 entries, 0 to 9133
Data columns (total 22 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   State                          9134 non-null   object 
 1   Customer Lifetime Value        9134 non-null   float64
 2   Response                       9134 non-null   object 
 3   Coverage                       9134 non-null   object 
 4   Education                      9134 non-null   object 
 5   EmploymentStatus               9134 non-null   object 
 6   Gender                         9134 non-null   object 
 7   Income                         9134 non-null   int64  
 8   Location Code                  9134 non-null   object 
 9   Marital Status                 9134 non-null   object 
 10  Monthly Premium Auto           9134 non-null   int64  
 11  Months Since Last Claim        9134 non-null   int64  
 12  Months Since Policy Inception  9134 non-null   int64  
 13  Number of Open Complaints      9134 non-null   int64  
 14  Number of Policies             9134 non-null   int64  
 15  Policy Type                    9134 non-null   object 
 16  Policy                         9134 non-null   object 
 17  Renew Offer Type               9134 non-null   object 
 18  Sales Channel                  9134 non-null   object 
 19  Total Claim Amount             9134 non-null   float64
 20  Vehicle Class                  9134 non-null   object 
 21  Vehicle Size                   9134 non-null   object 
dtypes: float64(2), int64(6), object(14)
memory usage: 1.5+ MB
```

Data Transformation using label encoder(df)

In [134…
```
##Data processing fns
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [135…
```
data["State"]=le.fit_transform(data["State"])
```

In [136…
```
data["Response"]=le.fit_transform(data["Response"])
```

In [137…
```
data["Coverage"]=le.fit_transform(data["Coverage"])
```

In [138…
```
data["Education"]=le.fit_transform(data["Education"])
```

In [139…
```
data["EmploymentStatus"]=le.fit_transform(data["EmploymentStatus"])
```

In [140…
```
data["Location Code"]=le.fit_transform(data["Location Code"])
```

```
In [141… data["Marital Status"]=le.fit_transform(data["Marital Status"])
```

```
In [142… data["Policy Type"]=le.fit_transform(data["Policy Type"])
```

```
In [143… data["Policy"]=le.fit_transform(data["Policy"])
```

```
In [144… data["Renew Offer Type"]=le.fit_transform(data["Renew Offer Type"])
```

```
In [145… data["Sales Channel"]=le.fit_transform(data["Sales Channel"])
```

```
In [146… data["Vehicle Class"]=le.fit_transform(data["Vehicle Class"])
```

```
In [147… data["Vehicle Size"]=le.fit_transform(data["Vehicle Size"])
```

```
In [148… data["Gender"]=le.fit_transform(data["Gender"])
```

```
In [149… data.dtypes
```

```
Out[149]: State                              int32
          Customer Lifetime Value          float64
          Response                           int32
          Coverage                           int32
          Education                          int32
          EmploymentStatus                   int32
          Gender                             int32
          Income                             int64
          Location Code                      int32
          Marital Status                     int32
          Monthly Premium Auto               int64
          Months Since Last Claim            int64
          Months Since Policy Inception      int64
          Number of Open Complaints          int64
          Number of Policies                 int64
          Policy Type                        int32
          Policy                             int32
          Renew Offer Type                   int32
          Sales Channel                      int32
          Total Claim Amount               float64
          Vehicle Class                      int32
          Vehicle Size                       int32
          dtype: object
```

```
In [150… # lets build our linear model
         # independant variables
         X = data.drop(['Customer Lifetime Value'], axis=1)
         # the dependent variable
         Y = data[['Customer Lifetime Value']]
```

```
In [151… # Split x and y into training and test set in 70:30 ratio

         X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30,rar
```

Loading [MathJax]/extensions/Safe.js

In [152…
```
# Data modelling with outliers and without outliers is also a good approach
```

In [153…
```
model_2 = LinearRegression()
model_2.fit(X_train, Y_train)
```

Out[153]:
```
▼ LinearRegression
LinearRegression()
```

In [154…
```
model_2.score(X_train, Y_train)
```

Out[154]: 0.17578211591653015

In [155…
```
model_2.score(X_test, Y_test)
```

Out[155]: 0.13189609072534558

In [156…
```
y_pred8=model_2.predict(X_test)
```

In [157…
```
print(sqrt(mean_squared_error(Y_test,y_pred8)))
```

6155.006339586745

In [158…
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model

poly = PolynomialFeatures(degree=2, interaction_only=True)
X_train2 = poly.fit_transform(X_train)
X_test2 = poly.fit_transform(X_test)

poly_model2 = linear_model.LinearRegression()

poly_model2.fit(X_train2, Y_train)

#In sample (training) R^2 will always improve with the number of variables!
print(poly_model2.score(X_train2, Y_train))
```

0.22045664759733052

In [159…
```
#Out off sample (testing) R^2 is our measure of sucess and does improve
print(poly_model2.score(X_test2, Y_test))
```

0.12162748115414046

In [160…
```
from sklearn.svm import SVR
```

In [161…
```
svr3=SVR()
svr3.fit(X_train, Y_train)
svr3.score(X_test,Y_test)
```

Out[161]: -0.10039687863536528

In [162…
```
print("Support Vector Regressor Accuracy : {:.2f}%".format(svr3.score(X_test
```

```
Support Vector Regressor Accuracy : -10.04%
```

In [163…
```
svr4=SVR(
    kernel='rbf',
    degree=3,
    gamma='auto',
    coef0=0.0,
    tol=0.001,
    C=1.0,
    epsilon=0.1,
    shrinking=True,
    cache_size=200,
    verbose=False,
    max_iter=-1,
)
```

In [164…
```
svr4.fit(X_train, Y_train)
svr4.score(X_test,Y_test)
```

Out[164]:
```
-0.09944980640198375
```

In [165…
```
svr4.fit(X_train, Y_train)
print("SVR Accuracy with Auto : {:.2f}%".format(svr4.score(X_test,Y_test)*10
```

```
SVR Accuracy with Auto : -9.94%
```

In [166…
```
y_pred9=svr4.predict(X_test)
```

In [167…
```
print(sqrt(mean_squared_error(Y_test,y_pred9)))
```

```
6926.763378370393
```

In [168…
```
# {'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}
```

In [169…
```
svr2=SVR(
    kernel='linear',
    degree=3,
    gamma='scale',
    coef0=0.0,
    tol=0.001,
    C=1.0,
    epsilon=0.1,
    shrinking=True,
    cache_size=200,
    verbose=False,
    max_iter=-1,
)
```

In [170…
```
#svr2.fit(x_train, y_train)
#svr2.score(x_test,y_test)
```

In [171…
```
#svr2.fit(x_train, y_train)
#print("SVR Accuracy with Linear Kernel : {:.2f}%".format(svr2.score(x_test,
```

Decision Tree Regressor

In [172…
```
dtr2=DecisionTreeRegressor(max_depth=6)
dtr2.fit(X_train, Y_train)
```

Out[172]:
```
▼        DecisionTreeRegressor
DecisionTreeRegressor(max_depth=6)
```

In [173…
```
print("Decision Tree Regressor : {:.2f}%".format(dtr2.score(X_test,Y_test)*1
```

```
Decision Tree Regressor : 66.66%
```

In [174…
```
y_pred10=dtr2.predict(X_test)
```

In [175…
```
print(sqrt(mean_squared_error(Y_test,y_pred10)))
```

```
3814.2006361518047
```

In [176…
```
from sklearn import tree
plt.figure(figsize=(15,10))
tree.plot_tree(dtr2,filled=True)
```

Out[176]:

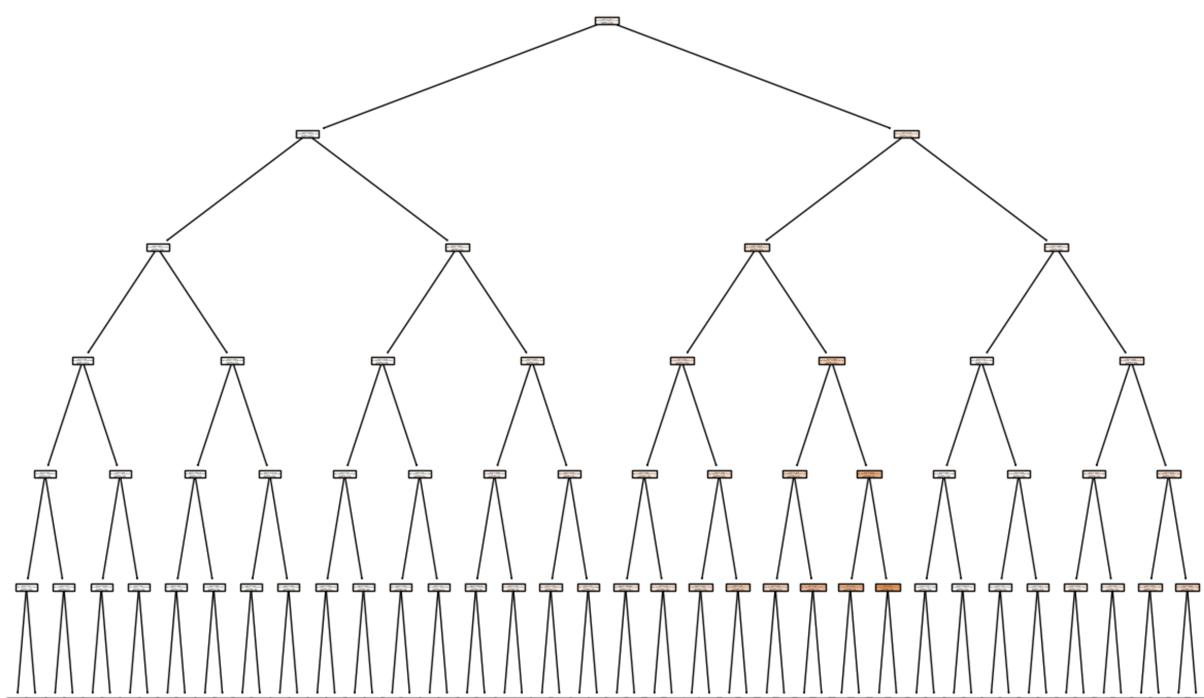
```
[Text(0.5, 0.9285714285714286, 'x[13] <= 1.5\nsquared_error = 48727636.976
\nsamples = 6393\nvalue = 8046.92'),
 Text(0.25, 0.7857142857142857, 'x[9] <= 124.5\nsquared_error = 1974859.52
1\nsamples = 2263\nvalue = 3580.027'),
 Text(0.125, 0.6428571428571429, 'x[9] <= 87.5\nsquared_error = 532494.138
\nsamples = 1966\nvalue = 3163.816'),
 Text(0.0625, 0.5, 'x[9] <= 72.5\nsquared_error = 91206.936\nsamples = 125
4\nvalue = 2684.768'),
 Text(0.03125, 0.35714285714285715, 'x[9] <= 67.5\nsquared_error = 38358.1
07\nsamples = 872\nvalue = 2542.577'),
 Text(0.015625, 0.21428571428571427, 'x[4] <= 2.5\nsquared_error = 24978.5
21\nsamples = 530\nvalue = 2446.996'),
 Text(0.0078125, 0.07142857142857142, 'squared_error = 9078.812\nsamples =
386\nvalue = 2519.389'),
 Text(0.0234375, 0.07142857142857142, 'squared_error = 15894.346\nsamples
= 144\nvalue = 2252.944'),
 Text(0.046875, 0.21428571428571427, 'x[4] <= 1.5\nsquared_error = 22994.3
38\nsamples = 342\nvalue = 2690.7'),
 Text(0.0390625, 0.07142857142857142, 'squared_error = 6585.647\nsamples =
232\nvalue = 2767.98'),
 Text(0.0546875, 0.07142857142857142, 'squared_error = 18440.75\nsamples =
110\nvalue = 2527.711'),
 Text(0.09375, 0.35714285714285715, 'x[9] <= 78.5\nsquared_error = 60340.0
27\nsamples = 382\nvalue = 3009.35'),
 Text(0.078125, 0.21428571428571427, 'x[6] <= 5156.0\nsquared_error = 2887
5.548\nsamples = 192\nvalue = 2855.365'),
 Text(0.0703125, 0.07142857142857142, 'squared_error = 24833.157\nsamples
= 47\nvalue = 2635.77'),
 Text(0.0859375, 0.07142857142857142, 'squared_error = 9488.826\nsamples =
145\nvalue = 2926.544'),
 Text(0.109375, 0.21428571428571427, 'x[4] <= 2.5\nsquared_error = 43961.3
95\nsamples = 190\nvalue = 3164.956'),
 Text(0.1015625, 0.07142857142857142, 'squared_error = 15823.831\nsamples
= 139\nvalue = 3263.092'),
 Text(0.1171875, 0.07142857142857142, 'squared_error = 22861.93\nsamples =
51\nvalue = 2897.487'),
 Text(0.1875, 0.5, 'x[9] <= 105.5\nsquared_error = 193665.932\nsamples = 7
12\nvalue = 4007.533'),
 Text(0.15625, 0.35714285714285715, 'x[6] <= 16936.5\nsquared_error = 7862
1.967\nsamples = 378\nvalue = 3701.224'),
 Text(0.140625, 0.21428571428571427, 'x[9] <= 94.5\nsquared_error = 62501.
815\nsamples = 114\nvalue = 3468.946'),
 Text(0.1328125, 0.07142857142857142, 'squared_error = 32404.997\nsamples
= 37\nvalue = 3260.997'),
 Text(0.1484375, 0.07142857142857142, 'squared_error = 46200.045\nsamples
= 77\nvalue = 3568.87'),
 Text(0.171875, 0.21428571428571427, 'x[9] <= 98.5\nsquared_error = 52224.
661\nsamples = 264\nvalue = 3801.526'),
 Text(0.1640625, 0.07142857142857142, 'squared_error = 23345.736\nsamples
= 153\nvalue = 3652.234'),
 Text(0.1796875, 0.07142857142857142, 'squared_error = 18964.222\nsamples
= 111\nvalue = 4007.306'),
 Text(0.21875, 0.35714285714285715, 'x[6] <= 11716.0\nsquared_error = 9750
6.043\nsamples = 334\nvalue = 4354.194'),
 Text(0.203125, 0.21428571428571427, 'x[9] <= 113.5\nsquared_error = 9115
[illegible]es = 83\nvalue = 3990.861'),
```

```
 Text(0.1953125, 0.07142857142857142, 'squared_error = 47669.993\nsamples
= 43\nvalue = 3803.917'),
 Text(0.2109375, 0.07142857142857142, 'squared_error = 59945.541\nsamples
= 40\nvalue = 4191.827'),
 Text(0.234375, 0.21428571428571427, 'x[9] <= 113.5\nsquared_error = 4151
8.036\nsamples = 251\nvalue = 4474.34'),
 Text(0.2265625, 0.07142857142857142, 'squared_error = 15632.96\nsamples =
134\nvalue = 4329.657'),
 Text(0.2421875, 0.07142857142857142, 'squared_error = 19731.14\nsamples =
117\nvalue = 4640.046'),
 Text(0.375, 0.6428571428571429, 'x[9] <= 169.0\nsquared_error = 2785234.7
14\nsamples = 297\nvalue = 6335.148'),
 Text(0.3125, 0.5, 'x[9] <= 145.0\nsquared_error = 240850.491\nsamples = 1
80\nvalue = 5235.195'),
 Text(0.28125, 0.35714285714285715, 'x[4] <= 1.5\nsquared_error = 104459.5
31\nsamples = 149\nvalue = 5074.129'),
 Text(0.265625, 0.21428571428571427, 'x[9] <= 134.5\nsquared_error = 4536
8.008\nsamples = 93\nvalue = 5258.943'),
 Text(0.2578125, 0.07142857142857142, 'squared_error = 15865.356\nsamples
= 52\nvalue = 5103.113'),
 Text(0.2734375, 0.07142857142857142, 'squared_error = 12927.586\nsamples
= 41\nvalue = 5456.58'),
 Text(0.296875, 0.21428571428571427, 'x[9] <= 129.5\nsquared_error = 5166
9.495\nsamples = 56\nvalue = 4767.208'),
 Text(0.2890625, 0.07142857142857142, 'squared_error = 48895.698\nsamples
= 18\nvalue = 4555.475'),
 Text(0.3046875, 0.07142857142857142, 'squared_error = 21688.76\nsamples =
38\nvalue = 4867.502'),
 Text(0.34375, 0.35714285714285715, 'x[4] <= 1.5\nsquared_error = 172402.8
75\nsamples = 31\nvalue = 6009.349'),
 Text(0.328125, 0.21428571428571427, 'x[9] <= 162.5\nsquared_error = 7788
4.762\nsamples = 20\nvalue = 6242.564'),
 Text(0.3203125, 0.07142857142857142, 'squared_error = 16685.323\nsamples
= 13\nvalue = 6079.75'),
 Text(0.3359375, 0.07142857142857142, 'squared_error = 50883.834\nsamples
= 7\nvalue = 6544.933'),
 Text(0.359375, 0.21428571428571427, 'x[3] <= 2.0\nsquared_error = 65566.0
58\nsamples = 11\nvalue = 5585.322'),
 Text(0.3515625, 0.07142857142857142, 'squared_error = 29263.591\nsamples
= 4\nvalue = 5319.325'),
 Text(0.3671875, 0.07142857142857142, 'squared_error = 22775.611\nsamples
= 7\nvalue = 5737.321'),
 Text(0.4375, 0.5, 'x[9] <= 227.5\nsquared_error = 1974629.623\nsamples =
117\nvalue = 8027.384'),
 Text(0.40625, 0.35714285714285715, 'x[9] <= 208.0\nsquared_error = 37631
5.371\nsamples = 91\nvalue = 7372.907'),
 Text(0.390625, 0.21428571428571427, 'x[6] <= 22365.5\nsquared_error = 222
511.0\nsamples = 78\nvalue = 7205.089'),
 Text(0.3828125, 0.07142857142857142, 'squared_error = 169003.107\nsamples
= 21\nvalue = 6686.609'),
 Text(0.3984375, 0.07142857142857142, 'squared_error = 106696.409\nsamples
= 57\nvalue = 7396.108'),
 Text(0.421875, 0.21428571428571427, 'x[12] <= 1.5\nsquared_error = 11630
6.174\nsamples = 13\nvalue = 8379.812'),
 Text(0.4140625, 0.07142857142857142, 'squared_error = 37622.82\nsamples =
Loading [MathJax]/extensions/Safe.js 8541.706'),
```

```
 Text(0.4296875, 0.07142857142857142, 'squared_error = 0.0\nsamples = 3\nv
alue = 7840.166'),
 Text(0.46875, 0.35714285714285715, 'x[9] <= 275.5\nsquared_error = 82236
4.802\nsamples = 26\nvalue = 10318.055'),
 Text(0.453125, 0.21428571428571427, 'x[11] <= 42.0\nsquared_error = 38938
8.649\nsamples = 14\nvalue = 9613.166'),
 Text(0.4453125, 0.07142857142857142, 'squared_error = 117530.427\nsamples
= 5\nvalue = 10266.112'),
 Text(0.4609375, 0.07142857142857142, 'squared_error = 171980.373\nsamples
= 9\nvalue = 9250.419'),
 Text(0.484375, 0.21428571428571427, 'x[1] <= 0.5\nsquared_error = 71531.8
88\nsamples = 12\nvalue = 11140.425'),
 Text(0.4765625, 0.07142857142857142, 'squared_error = 9656.16\nsamples =
3\nvalue = 11594.539'),
 Text(0.4921875, 0.07142857142857142, 'squared_error = 503.858\nsamples =
9\nvalue = 10989.053'),
 Text(0.75, 0.7857142857142857, 'x[13] <= 2.5\nsquared_error = 57421553.95
6\nsamples = 4130\nvalue = 10494.517'),
 Text(0.625, 0.6428571428571429, 'x[9] <= 102.5\nsquared_error = 89924250.
072\nsamples = 1626\nvalue = 15764.183'),
 Text(0.5625, 0.5, 'x[9] <= 81.5\nsquared_error = 37968454.523\nsamples =
1094\nvalue = 12722.563'),
 Text(0.53125, 0.35714285714285715, 'x[9] <= 69.5\nsquared_error = 3089323
0.242\nsamples = 788\nvalue = 11718.551'),
 Text(0.515625, 0.21428571428571427, 'x[18] <= 681.668\nsquared_error = 28
336818.596\nsamples = 489\nvalue = 11125.849'),
 Text(0.5078125, 0.07142857142857142, 'squared_error = 28343997.911\nsampl
es = 480\nvalue = 11034.438'),
 Text(0.5234375, 0.07142857142857142, 'squared_error = 3740063.355\nsample
s = 9\nvalue = 16001.113'),
 Text(0.546875, 0.21428571428571427, 'x[12] <= 2.5\nsquared_error = 335599
78.816\nsamples = 299\nvalue = 12687.887'),
 Text(0.5390625, 0.07142857142857142, 'squared_error = 34592789.413\nsampl
es = 276\nvalue = 13041.478'),
 Text(0.5546875, 0.07142857142857142, 'squared_error = 1661998.523\nsample
s = 23\nvalue = 8444.783'),
 Text(0.59375, 0.35714285714285715, 'x[10] <= 33.5\nsquared_error = 469076
63.91\nsamples = 306\nvalue = 15308.057'),
 Text(0.578125, 0.21428571428571427, 'x[18] <= 4.41\nsquared_error = 42835
224.969\nsamples = 290\nvalue = 14972.809'),
 Text(0.5703125, 0.07142857142857142, 'squared_error = 0.0\nsamples = 1\nv
alue = 35186.256'),
 Text(0.5859375, 0.07142857142857142, 'squared_error = 41564768.176\nsampl
es = 289\nvalue = 14902.866'),
 Text(0.609375, 0.21428571428571427, 'x[19] <= 1.5\nsquared_error = 817611
11.214\nsamples = 16\nvalue = 21384.439'),
 Text(0.6015625, 0.07142857142857142, 'squared_error = 18677669.531\nsampl
es = 7\nvalue = 13457.116'),
 Text(0.6171875, 0.07142857142857142, 'squared_error = 43932760.099\nsampl
es = 9\nvalue = 27550.134'),
 Text(0.6875, 0.5, 'x[9] <= 160.5\nsquared_error = 138619033.751\nsamples
= 532\nvalue = 22018.944'),
 Text(0.65625, 0.35714285714285715, 'x[10] <= 33.5\nsquared_error = 866108
09.734\nsamples = 449\nvalue = 19829.704'),
 Text(0.640625, 0.21428571428571427, 'x[1] <= 0.5\nsquared_error = 7930036
[illegible]es = 432\nvalue = 19461.54'),
```

```
 Text(0.6328125, 0.07142857142857142, 'squared_error = 84257648.135\nsampl
es = 363\nvalue = 20128.262'),
 Text(0.6484375, 0.07142857142857142, 'squared_error = 38579359.728\nsampl
es = 69\nvalue = 15954.002'),
 Text(0.671875, 0.21428571428571427, 'x[2] <= 0.5\nsquared_error = 1814085
55.612\nsamples = 17\nvalue = 29185.406'),
 Text(0.6640625, 0.07142857142857142, 'squared_error = 76554304.026\nsampl
es = 8\nvalue = 21023.093'),
 Text(0.6796875, 0.07142857142857142, 'squared_error = 162750899.514\nsamp
les = 9\nvalue = 36440.796'),
 Text(0.71875, 0.35714285714285715, 'x[9] <= 215.5\nsquared_error = 253780
922.204\nsamples = 83\nvalue = 33861.944'),
 Text(0.703125, 0.21428571428571427, 'x[3] <= 2.0\nsquared_error = 2076705
76.953\nsamples = 61\nvalue = 31033.325'),
 Text(0.6953125, 0.07142857142857142, 'squared_error = 244529254.718\nsamp
les = 43\nvalue = 33902.618'),
 Text(0.7109375, 0.07142857142857142, 'squared_error = 52968842.223\nsampl
es = 18\nvalue = 24178.903'),
 Text(0.734375, 0.21428571428571427, 'x[3] <= 2.5\nsquared_error = 2979350
71.223\nsamples = 22\nvalue = 41704.931'),
 Text(0.7265625, 0.07142857142857142, 'squared_error = 130693253.267\nsamp
les = 13\nvalue = 33699.016'),
 Text(0.7421875, 0.07142857142857142, 'squared_error = 313197017.351\nsamp
les = 9\nvalue = 53269.03'),
 Text(0.875, 0.6428571428571429, 'x[9] <= 100.5\nsquared_error = 6573708.7
06\nsamples = 2504\nvalue = 7072.601'),
 Text(0.8125, 0.5, 'x[9] <= 77.5\nsquared_error = 782173.697\nsamples = 16
49\nvalue = 5671.545'),
 Text(0.78125, 0.35714285714285715, 'x[9] <= 69.5\nsquared_error = 213434.
283\nsamples = 1145\nvalue = 5197.868'),
 Text(0.765625, 0.21428571428571427, 'x[4] <= 1.5\nsquared_error = 131518.
209\nsamples = 700\nvalue = 4974.696'),
 Text(0.7578125, 0.07142857142857142, 'squared_error = 61478.459\nsamples
= 479\nvalue = 5141.23'),
 Text(0.7734375, 0.07142857142857142, 'squared_error = 92928.958\nsamples
= 221\nvalue = 4613.746'),
 Text(0.796875, 0.21428571428571427, 'x[6] <= 15127.0\nsquared_error = 140
702.394\nsamples = 445\nvalue = 5548.927'),
 Text(0.7890625, 0.07142857142857142, 'squared_error = 112577.665\nsamples
= 134\nvalue = 5127.029'),
 Text(0.8046875, 0.07142857142857142, 'squared_error = 43081.93\nsamples =
311\nvalue = 5730.709'),
 Text(0.84375, 0.35714285714285715, 'x[9] <= 87.5\nsquared_error = 406508.
989\nsamples = 504\nvalue = 6747.655'),
 Text(0.828125, 0.21428571428571427, 'x[4] <= 1.5\nsquared_error = 162123.
572\nsamples = 251\nvalue = 6318.427'),
 Text(0.8203125, 0.07142857142857142, 'squared_error = 62003.596\nsamples
= 166\nvalue = 6515.867'),
 Text(0.8359375, 0.07142857142857142, 'squared_error = 132843.273\nsamples
= 85\nvalue = 5932.839'),
 Text(0.859375, 0.21428571428571427, 'x[4] <= 2.5\nsquared_error = 284846.
559\nsamples = 253\nvalue = 7173.49'),
 Text(0.8515625, 0.07142857142857142, 'squared_error = 121694.916\nsamples
= 179\nvalue = 7417.57'),
 Text(0.8671875, 0.07142857142857142, 'squared_error = 186808.478\nsamples
= 6583.082'),
```

```
 Text(0.9375, 0.5, 'x[9] <= 155.5\nsquared_error = 6656068.632\nsamples =
855\nvalue = 9774.756'),
 Text(0.90625, 0.35714285714285715, 'x[9] <= 122.5\nsquared_error = 86704
7.829\nsamples = 725\nvalue = 8811.681'),
 Text(0.890625, 0.21428571428571427, 'x[9] <= 112.5\nsquared_error = 37757
7.964\nsamples = 573\nvalue = 8462.172'),
 Text(0.8828125, 0.07142857142857142, 'squared_error = 250850.905\nsamples
= 346\nvalue = 8161.201'),
 Text(0.8984375, 0.07142857142857142, 'squared_error = 222218.238\nsamples
= 227\nvalue = 8920.921'),
 Text(0.921875, 0.21428571428571427, 'x[4] <= 2.5\nsquared_error = 515763.
742\nsamples = 152\nvalue = 10129.239'),
 Text(0.9140625, 0.07142857142857142, 'squared_error = 272640.951\nsamples
= 103\nvalue = 10467.58'),
 Text(0.9296875, 0.07142857142857142, 'squared_error = 280370.086\nsamples
= 49\nvalue = 9418.031'),
 Text(0.96875, 0.35714285714285715, 'x[9] <= 211.5\nsquared_error = 492071
6.349\nsamples = 130\nvalue = 15145.753'),
 Text(0.953125, 0.21428571428571427, 'x[9] <= 186.5\nsquared_error = 14699
15.596\nsamples = 99\nvalue = 14203.034'),
 Text(0.9453125, 0.07142857142857142, 'squared_error = 891314.58\nsamples
= 43\nvalue = 13329.634'),
 Text(0.9609375, 0.07142857142857142, 'squared_error = 878687.174\nsamples
= 56\nvalue = 14873.682'),
 Text(0.984375, 0.21428571428571427, 'x[9] <= 264.5\nsquared_error = 40390
40.093\nsamples = 31\nvalue = 18156.369'),
 Text(0.9765625, 0.07142857142857142, 'squared_error = 2022282.841\nsample
s = 28\nvalue = 17689.297'),
 Text(0.9921875, 0.07142857142857142, 'squared_error = 1822102.395\nsample
s = 3\nvalue = 22515.713')]
```



Random Forest Regressor

In [177…
```
rf2=RandomForestRegressor(n_estimators=1000,random_state=1,max_depth=6)
```

In [178…
```
rf2.fit(X_train, Y_train)
print("Random Forest Regressor : {:.2f}%".format(rf2.score(X_test,Y_test)*10
```

```
Random Forest Regressor : 69.06%
```

In [179…
```
y_pred11=rf2.predict(X_test)
```

In [180…
```
print(sqrt(mean_squared_error(Y_test,y_pred11)))
```

```
3674.54221209146
```

Bagging

In [181…
```
bgr2=BaggingRegressor(n_estimators=23)
```

In [182…
```
bgr2.fit(X_train, Y_train)
```

Out[182]:
```
▼          BaggingRegressor
BaggingRegressor(n_estimators=23)
```

In [183…
```
print(bgr2.score(X_train,Y_train))
print(bgr2.score(X_test,Y_test))
```

```
0.9506908154277011
0.6965142386442115
```

In [184…
```
print("Bagging Regressor : {:.2f}%".format(bgr2.score(X_test,Y_test)*100))
```

```
Bagging Regressor : 69.65%
```

In [185…
```
y_pred12=bgr2.predict(X_test)
```

In [186…
```
print(sqrt(mean_squared_error(Y_test,y_pred12)))
```

```
3639.2493126832446
```

Here model is overfitted

XG-Boost

In [187…
```
xgbr2=xgboost.XGBRFRegressor()
```

In [188…
```
xgbr2.fit(X_train, Y_train)
```

Out[188]:

▼ XGBRFRegressor

```
XGBRFRegressor(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bytree=None, devic
e=None,
               early_stopping_rounds=None, enable_categorical=Fals
e,
               eval_metric=None, feature_types=None, gamma=None,
               grow_policy=None, importance_type=None,
               interaction_constraints=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
```

In [189…
```
print(xgbr2.score(X_train,Y_train))
print(xgbr2.score(X_test,Y_test))
```

```
0.7374849309568463
0.6865858036209498
```

In [190…
```
print("Boosting Regressor : {:.2f}%".format(xgbr2.score(X_test,Y_test)*100))
```

```
Boosting Regressor : 68.66%
```

In [191…
```
y_pred13=xgbr2.predict(X_test)
```

In [192…
```
print(sqrt(mean_squared_error(Y_test,y_pred13)))
```

```
3698.2986653704056
```

Note: Here boosting regressor is the best on because train and test score differece is around 5 %

Adaboost

In [193…
```
abr2=AdaBoostRegressor()
```

In [194…
```
abr2.fit(X_train, Y_train)
```

Out[194]:

▼ AdaBoostRegressor

```
AdaBoostRegressor()
```

In [195…
```
print(abr2.score(X_train, Y_train))
print(abr2.score(X_test,Y_test))
```

```
0.26243245580663355
0.17269620289643184
```

In [196…
```
print("Adaboost Regressor : {:.2f}%".format(abr2.score(X_test,Y_test)*100))
```

```
Adaboost Regressor : 17.27%
```

In [197…
```
y_pred14=abr2.predict(X_test)
```

In [198…
```
print(sqrt(mean_squared_error(Y_test,y_pred14)))
```

```
6008.625788696474
```

Gradient Boost

In [199…
```
gbr2=GradientBoostingRegressor()
```

In [200…
```
gbr2.fit(X_train, Y_train)
```

Out[200]:
```
▼ GradientBoostingRegressor
GradientBoostingRegressor()
```

In [201…
```
print(gbr2.score(X_train, Y_train))
print(gbr2.score(X_test,Y_test))
```

```
0.7369601378817796
0.6786628528679675
```

In [202…
```
print("Gradient boost Regressor : {:.2f}%".format(gbr2.score(X_test,Y_test)*
```

```
Gradient boost Regressor : 67.87%
```

Note Gradient boosting regressor is also good for the model

In [203…
```
y_pred15=gbr2.predict(X_test)
```

In [204…
```
print(sqrt(mean_squared_error(Y_test,y_pred15)))
```

```
3744.7524659496426
```

In [ ]: