Team 2 Final Project - Health Insurance Cross Sell

Background information on the overall project:

Our client is an Insurance company that has provided Health Insurance to its customers now they need your help in building a model to predict whether the policyholders (customers) from past year will also be interested in Vehicle Insurance provided by the company.

Just like medical insurance, there is vehicle insurance where every year customer needs to pay a premium of certain amount to insurance provider company so that in case of unfortunate accident by the vehicle, the insurance provider company will provide a compensation (called 'sum assured') to the customer.

Building a model to predict whether a customer would be interested in Vehicle Insurance is extremely helpful for the company because it can then accordingly plan its communication strategy to reach out to those customers and optimise its business model and revenue.

Now, in order to predict, whether the customer would be interested in Vehicle insurance, you have information about demographics (gender, age, region code type), Vehicles (Vehicle Age, Damage), Policy (Premium, sourcing channel) etc.

Problem Statement:

Our client currently provides for Health Insurance, but they are unsure if it is possible to transition into the automobile insurance sector. In order to determine if this can be done, the insurance company has tasked our team with developing models that would be able to predict whether a current policy holder would be interested in adding vehicle insurance to their policy.

Technical Notebook

Import needed packages

In [1]:
```
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pylab as plt
import seaborn as sns
import statsmodels.api as sm
import imblearn
from imblearn.over_sampling import RandomOverSampler

from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.neighbors import NearestNeighbors, KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.metrics import accuracy_score, roc_auc_score, f1_score, auc
from sklearn.preprocessing import MinMaxScaler
from sklearn.neural_network import MLPClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier

from dmba import classificationSummary, gainsChart
from dmba import plotDecisionTree, exhaustive_search
from dmba import backward_elimination, forward_selection, stepwise_selection
from dmba import adjusted_r2_score, AIC_score, BIC_score

import warnings
warnings.filterwarnings('ignore')
```

Import the client provided datasets. They have already split the data into training and testing sets.

In [2]:
```
#import the training set and view the first few rows
train = pd.read_csv('C:/Users/Leonard/Desktop/insurance_train.csv')
train.head()
```

Out[2]:

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel
0	1	Male	44	1	28.0	0	> 2 Years	Yes	40454.0	26.0
1	2	Male	76	1	3.0	0	1-2 Year	No	33536.0	26.0
2	3	Male	47	1	28.0	0	> 2 Years	Yes	38294.0	26.0
3	4	Male	21	1	11.0	1	< 1 Year	No	28619.0	152.0
4	5	Female	29	1	41.0	1	< 1 Year	No	27496.0	152.0

The testing data will be held out so that we can test our models on unseen data. Therefore, all of our EDA and modeling will take place on the training data. We will use the entire training set to build models. Cross validation will be used for best results.

Exploratory Data Analysis

Initial Analysis

First, let us define the variables in our training set.

Variable	Definition
id	Unique ID for the customer
Gender	Gender of the customer
Age	Age of the customer
Driving_License	0 : Customer does not have DL, 1 : Customer already has DL
Region_Code	Unique code for the region of the customer
Previously_Insured	1 : Customer already has Vehicle Insurance, 0 : Customer doesn't have Vehicle Insurance
Vehicle_Age	Age of the Vehicle
Vehicle_Damage	1 : Customer got his/her vehicle damaged in the past. 0 : Customer didn't get his/her vehicle damaged in the past.
Annual_Premium	The amount customer needs to pay as premium in the year
Policy_Sales_Channel	Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc.
Vintage	Number of Days, Customer has been associated with the company
Response	1 : Customer is interested, 0 : Customer is not interested

Let's check for missing data. Missing data can impact analyses, so we want to be sure we do not have missing data, and if we do, handle that appropriately.

In [6]:
```
#check for missing data in the training set
train.isnull().sum()
```

Out[6]:
```
id                      0
Gender                  0
Age                     0
Driving_License         0
Region_Code             0
Previously_Insured      0
Vehicle_Age             0
Vehicle_Damage          0
Annual_Premium          0
Policy_Sales_Channel    0
Vintage                 0
Response                0
dtype: int64
```

Great! We have no missing data! Now let us see what kind of values our data can take on.

In [7]:
```
#determine the number of unique values for each column
dict=[]
for i in train.columns:
    print(f'{i} has {train[i].nunique()} unique values')
```

```
id has 381109 unique values
Gender has 2 unique values
Age has 66 unique values
Driving_License has 2 unique values
Region_Code has 53 unique values
Previously_Insured has 2 unique values
Vehicle_Age has 3 unique values
Vehicle_Damage has 2 unique values
Annual_Premium has 48838 unique values
Policy_Sales_Channel has 155 unique values
Vintage has 290 unique values
Response has 2 unique values
```

In [14]:
```
#determine the unique values for the lower count columns
print("Gender can take on:\n", train.Gender.value_counts(), "\n")
print("Driving_License can take on:\n", train.Driving_License.value_counts(), "\n")
print("Previously_Insured can take on:\n", train.Previously_Insured.value_counts(), "\n")
print("Vehicle_Age can take on:\n", train.Vehicle_Age.value_counts(), "\n")
print("Vehicle_Damage can take on:\n", train.Vehicle_Damage.value_counts(), "\n")
print("Response can take on:\n", train.Response.value_counts(), "\n")
```

```
Gender can take on:
 Male      206089
Female    175020
Name: Gender, dtype: int64

Driving_License can take on:
 1    380297
0       812
Name: Driving_License, dtype: int64

Previously_Insured can take on:
 0    206481
1    174628
Name: Previously_Insured, dtype: int64

Vehicle_Age can take on:
 1-2 Year     200316
< 1 Year     164786
> 2 Years     16007
Name: Vehicle_Age, dtype: int64

Vehicle_Damage can take on:
 Yes    192413
No     188696
Name: Vehicle_Damage, dtype: int64

Response can take on:
 0    334399
1     46710
Name: Response, dtype: int64
```

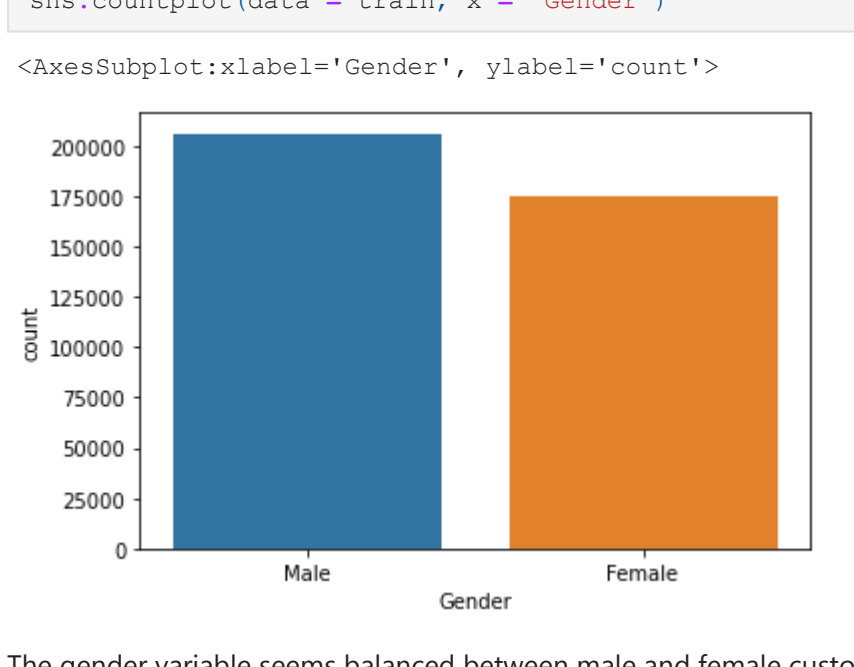
We make some observations about the columns here. First, the Response variable can take on 0 (not interested) or 1 (interested). We also see that this is very imbalanced. We also see that Gender is binary: either male or female. Driving_License is either 0 (no) or 1 (yes). This is also very imbalanced, but this is something we would expect since most insured individuals are licensed drivers. Previously_Insured is also binary: 0 (no) and 1 (yes). Vehicle_Age is categorized into three categories: < 1 year, 1-2 years, > 2 years. We note that this is an ordinal variable since the order of the values matter. Finally, Vehicle_Damage is either yes or no. The variables not mentioned here are continuous as they take on many values.

Distrubution Visualizations

We take a look at the frequency distribution of each of our variables

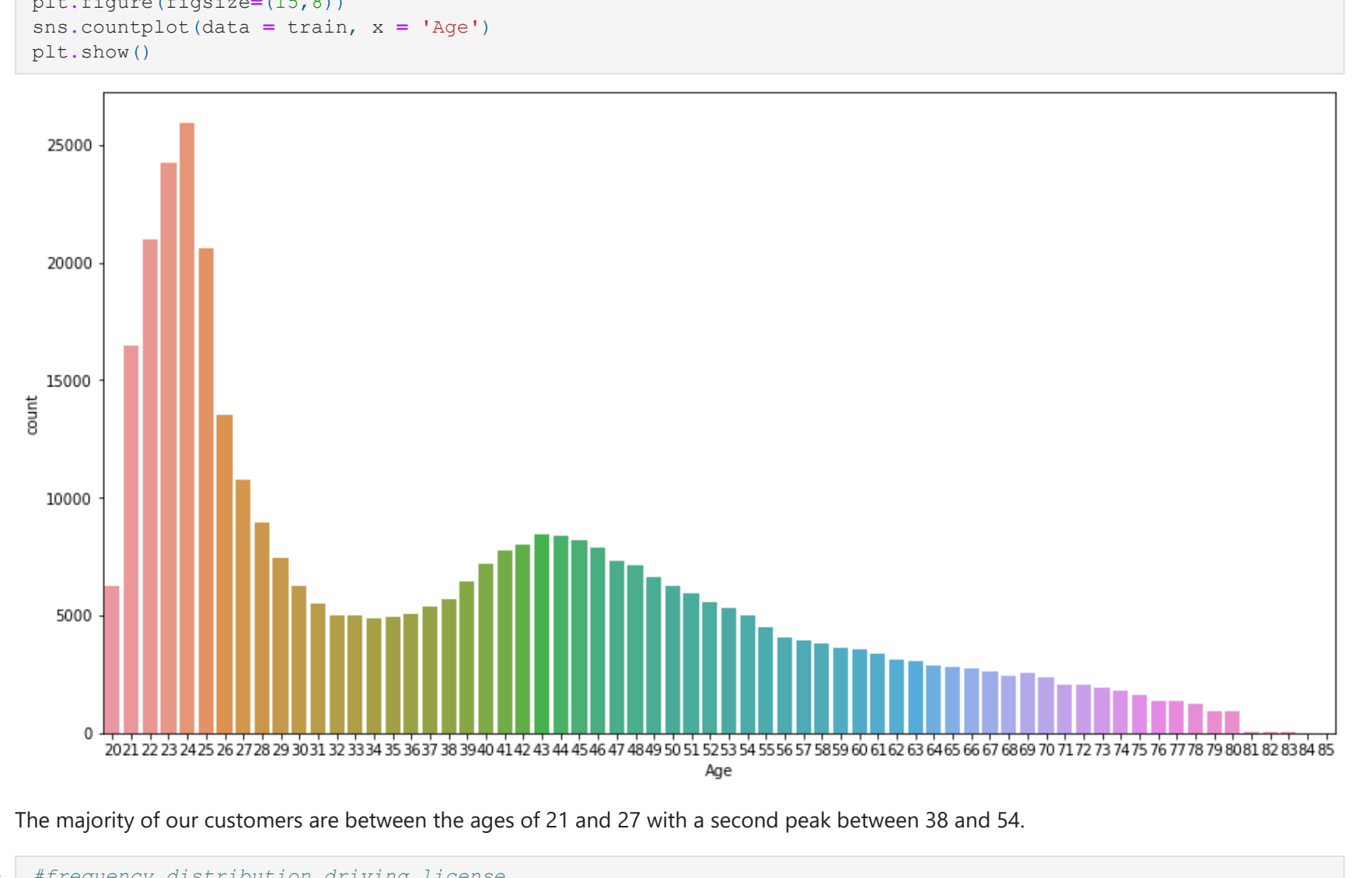
```
#frequency distribution gender
sns.countplot(data = train, x = 'Gender')
```

<AxesSubplot:xlabel='Gender', ylabel='count'>



The gender variable seems balanced between male and female customers

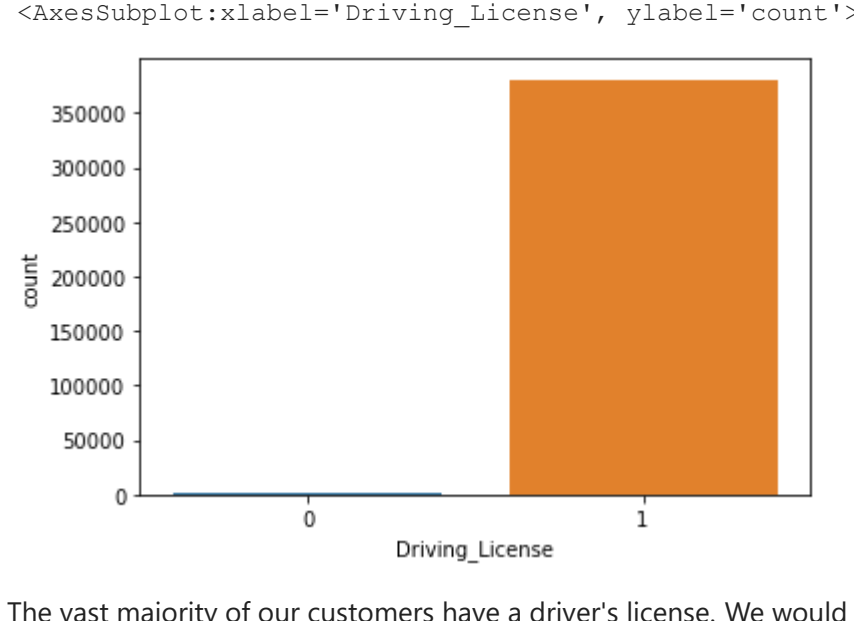
```
#frequency distribution age
plt.figure(figsize=(15,8))
sns.countplot(data = train, x = 'Age')
plt.show()
```



The majority of our customers are between the ages of 21 and 27 with a second peak between 38 and 54.

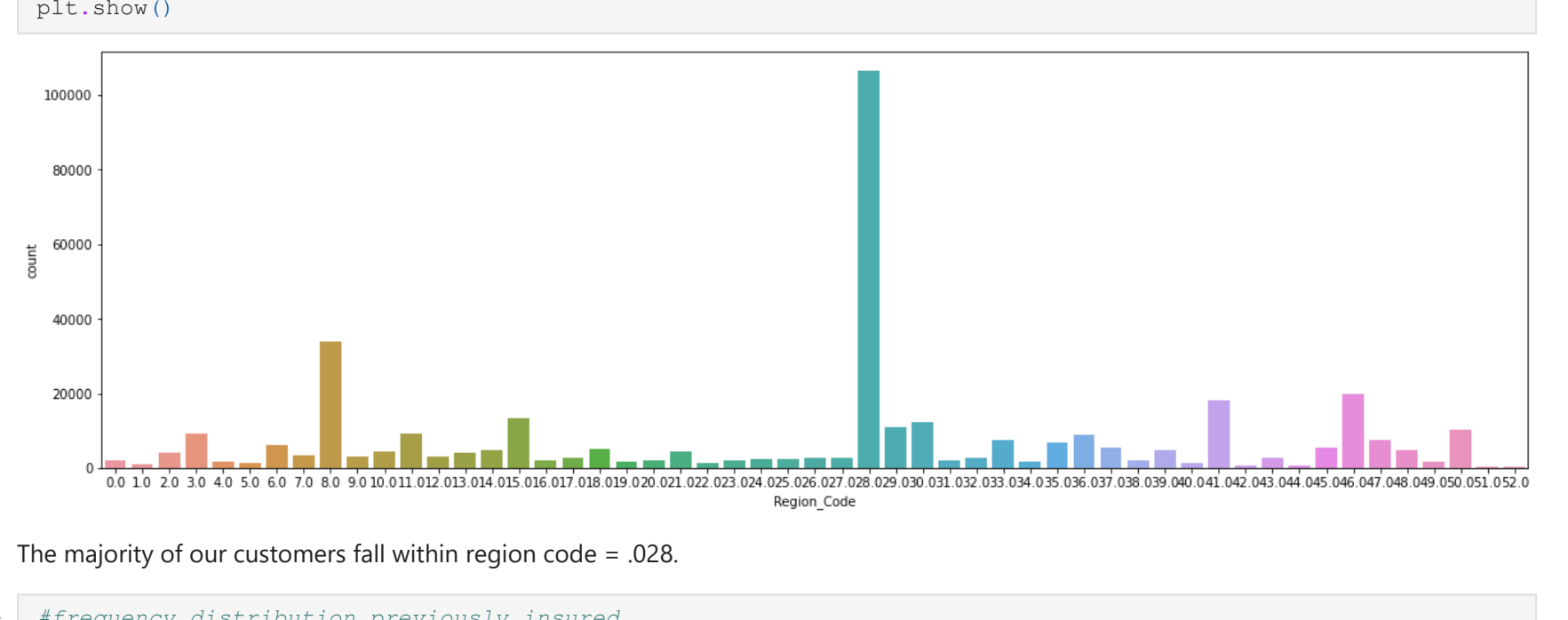
```
#frequency distribution driving_license
sns.countplot(data = train, x = 'Driving_License')
```

<AxesSubplot:xlabel='Driving_License', ylabel='count'>



The vast majority of our customers have a driver's license. We would expect this as customers who will be interested in car insurance should have a license.

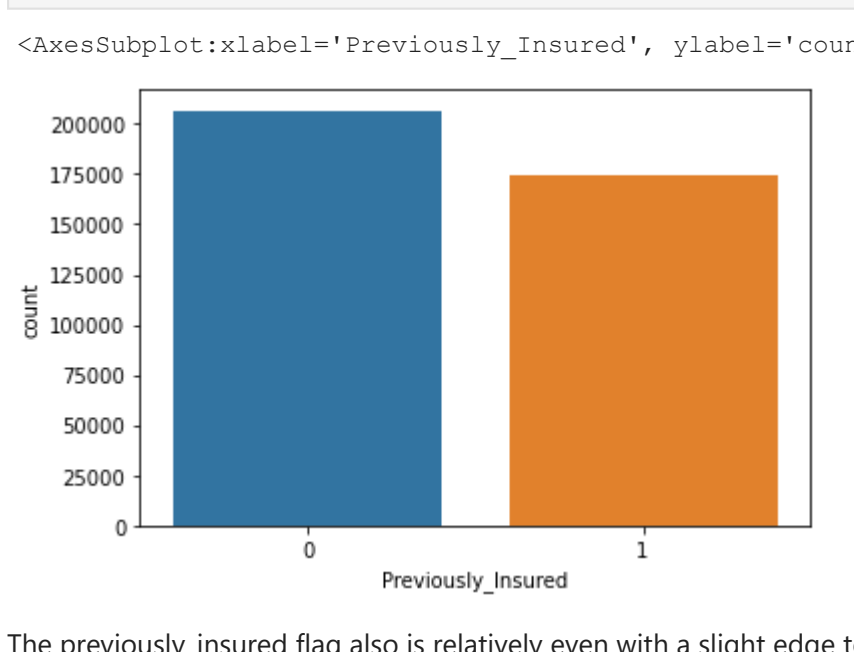
```
#frequency distribution region_code
plt.figure(figsize=(20,6))
sns.countplot(data = train, x = 'Region_Code')
plt.show()
```



The majority of our customers fall within region code = .028.

```
#frequency distribution previously_insured
sns.countplot(data = train, x = 'Previously_Insured')
```

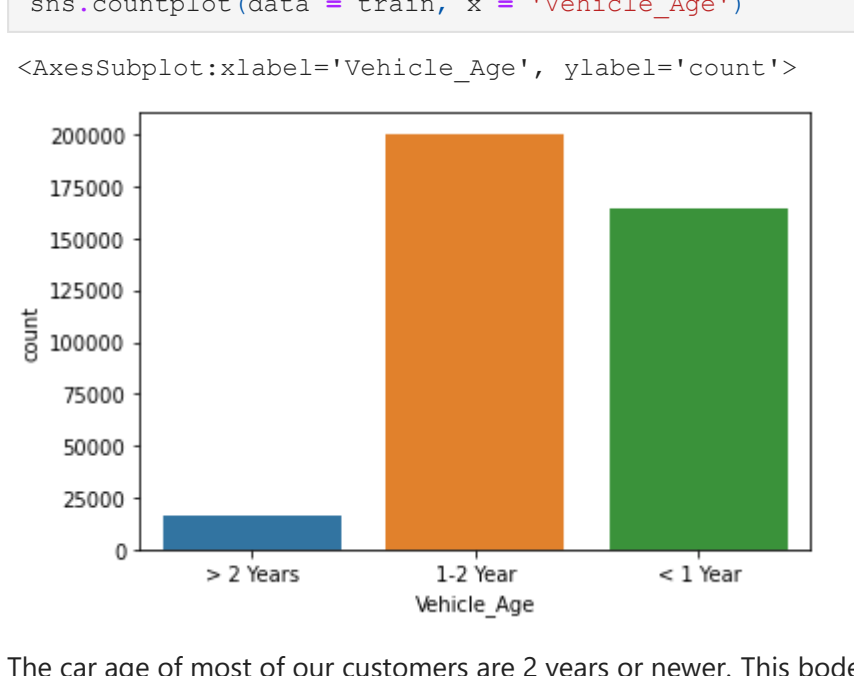
<AxesSubplot:xlabel='Previously_Insured', ylabel='count'>



The previously_insured flag also is relatively even with a slight edge to not previously insured.

```
#frequency distribution vehicle_age
sns.countplot(data = train, x = 'Vehicle_Age')
```

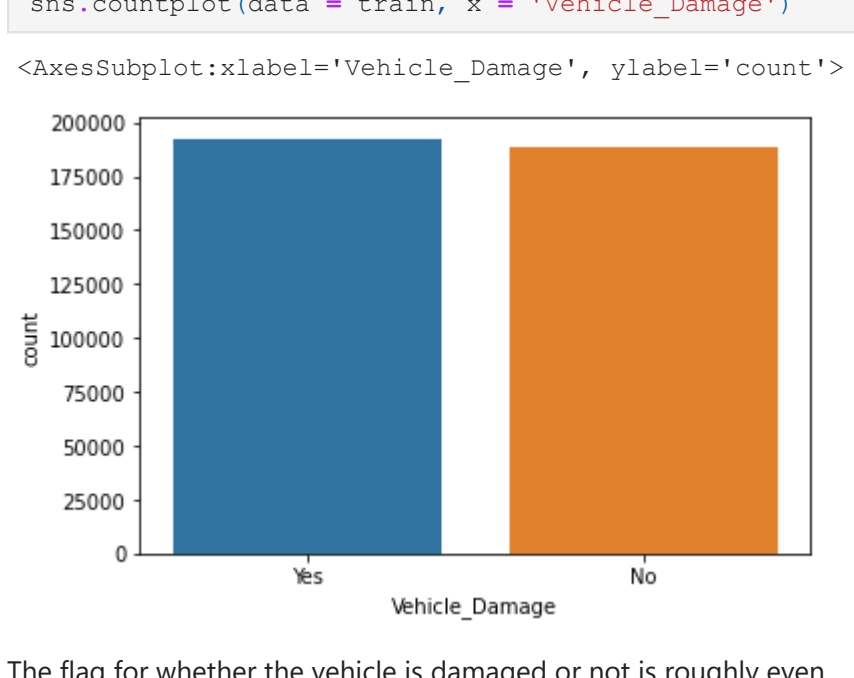
<AxesSubplot:xlabel='Vehicle_Age', ylabel='count'>



The car age of most of our customers are 2 years or newer. This bodes well for offering insurance as newer cars tend to be safer.

```
#frequency distribution vehicle damage
sns.countplot(data = train, x = 'Vehicle_Damage')
```

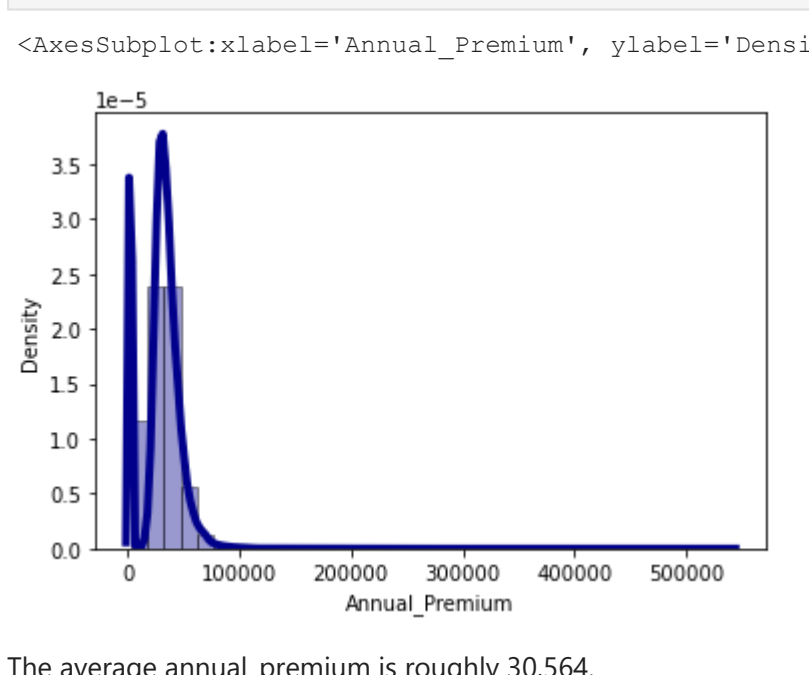
<AxesSubplot:xlabel='Vehicle_Damage', ylabel='count'>



The flag for whether the vehicle is damaged or not is roughly even.

In [38]:
```
#frequency distribution annual_premium
sns.distplot(train['Annual_Premium'], hist=True, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
```

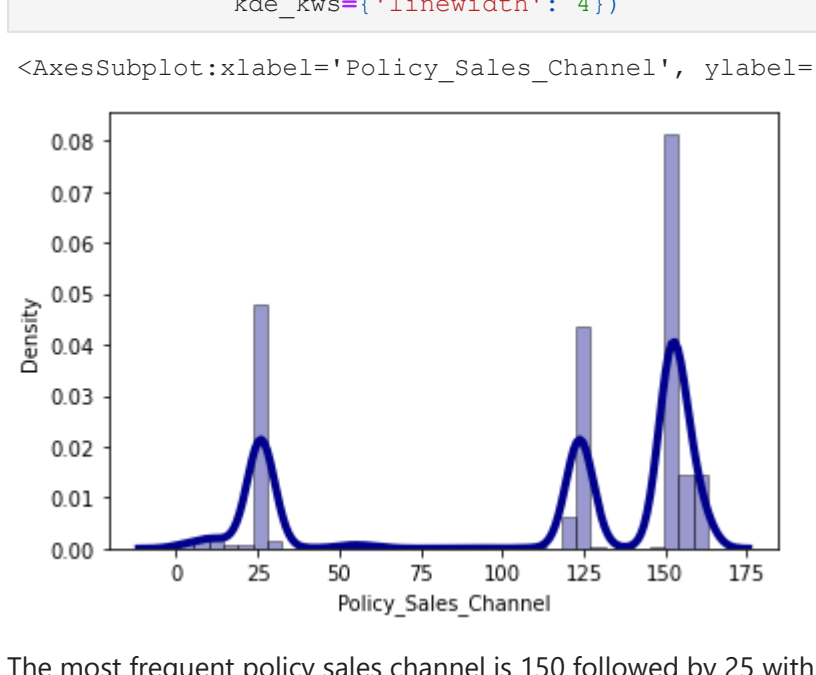
Out[38]: <AxesSubplot:xlabel='Annual_Premium', ylabel='Density'>



The average annual_premium is roughly 30,564.

In [49]:
```
#frequency distribution policy_sales_channel
sns.distplot(train['Policy_Sales_Channel'], hist=True, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
```

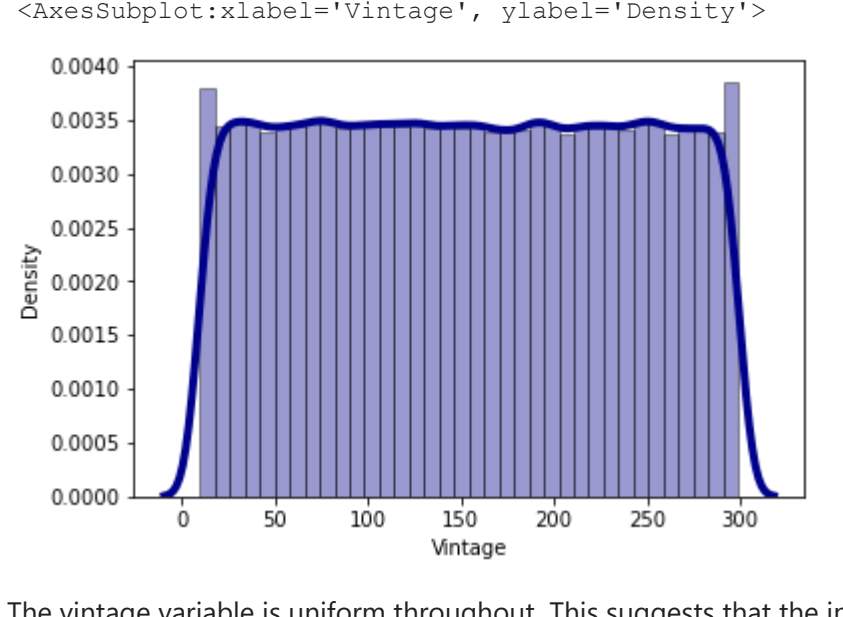
Out[49]: <AxesSubplot:xlabel='Policy_Sales_Channel', ylabel='Density'>



The most frequent policy sales channel is 150 followed by 25 with 125 coming in 3rd.

In [48]:
```
#frequency distribution vintage
sns.distplot(train['Vintage'], hist=True, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
```

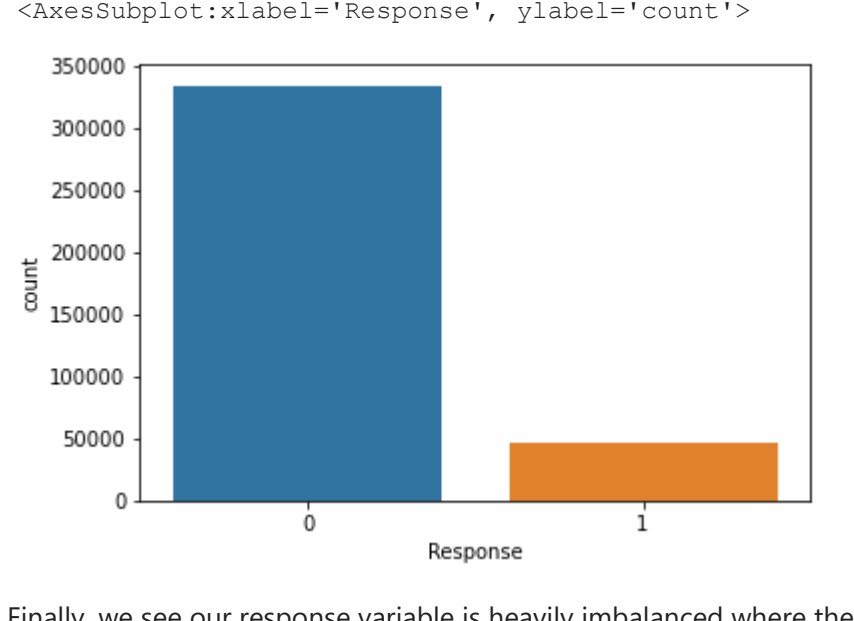
Out[48]: <AxesSubplot:xlabel='Vintage', ylabel='Density'>



The vintage variable is uniform throughout. This suggests that the insurance company is pretty good with getting new clients consistently each day.

In [50]:
```
#frequency distribution response
sns.countplot(data = train, x = 'Response')
```

Out[50]: <AxesSubplot:xlabel='Response', ylabel='count'>



Finally, we see our response variable is heavily imbalanced where the majority of our customers are not interested in car insurance. To handle this, we will need to use a technique called random oversampling.

Univariate Analysis

Many analyses won't work with string values. Here, we will map Gender, Vehicle_Age and Vehicle_Damage to integer placeholders. For Gender, we will map male to 0 and female to 1. For vehicle age, < 1 year will be 0, 1-2 years will be 1 and > 2 years will be 2. Finally, vehicle damage will be 0 for no and 1 for yes.

In [3]:
```
#map gender
train['Gender'] = train['Gender'].replace({'Male': 0, 'Female': 1})
train.head()
```

Out[3]:

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel
0	1	0	44	1	28.0	0	> 2 Years	Yes	40454.0	26.0
1	2	0	76	1	3.0	0	1-2 Year	No	33536.0	26.0
2	3	0	47	1	28.0	0	> 2 Years	Yes	38294.0	26.0
3	4	0	21	1	11.0	1	< 1 Year	No	28619.0	152.0
4	5	1	29	1	41.0	1	< 1 Year	No	27496.0	152.0

In [4]:
```
#map vehicle age
train['Vehicle_Age'] = train['Vehicle_Age'].replace({'< 1 Year': 0, '1-2 Year': 1, '> 2 Years': 2})
train.head()
```

Out[4]:

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel
0	1	0	44	1	28.0	0	2	Yes	40454.0	26.0
1	2	0	76	1	3.0	0	1	No	33536.0	26.0
2	3	0	47	1	28.0	0	2	Yes	38294.0	26.0
3	4	0	21	1	11.0	1	0	No	28619.0	152.0
4	5	1	29	1	41.0	1	0	No	27496.0	152.0

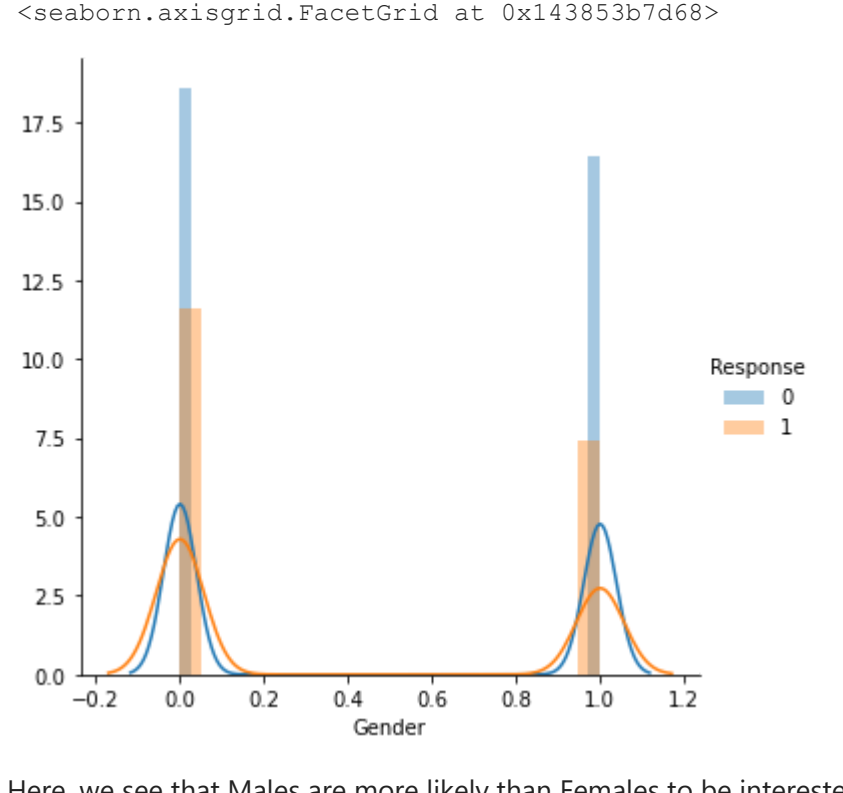
In [5]:
```
#map vehicle damage
train['Vehicle_Damage'] = train['Vehicle_Damage'].replace({'No': 0, 'Yes': 1})
train.head()
```

Out[5]:

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Channel
0	1	0	44	1	28.0	0	2	1	40454.0	26.0
1	2	0	76	1	3.0	0	1	0	33536.0	26.0
2	3	0	47	1	28.0	0	2	1	38294.0	26.0
3	4	0	21	1	11.0	1	0	0	28619.0	152.0
4	5	1	29	1	41.0	1	0	0	27496.0	152.0

In [26]:
```
#analysis for Gender
sns.FacetGrid(train, hue='Response', size = 5).map(sns.distplot, "Gender").add_legend()
```

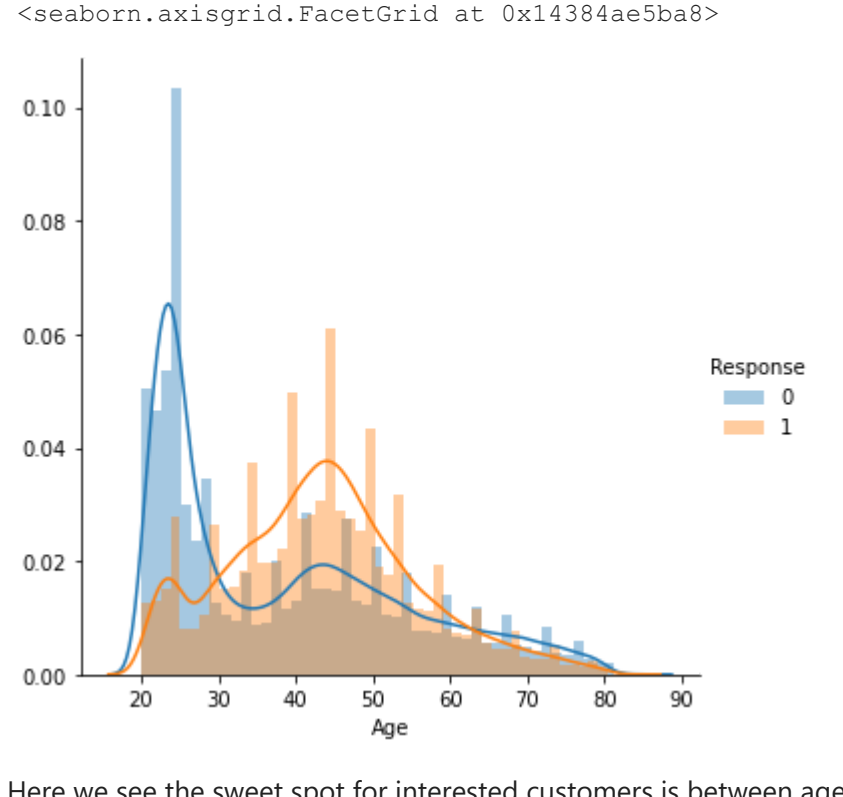
Out[26]: <seaborn.axisgrid.FacetGrid at 0x143853b7d68>



Here, we see that Males are more likely than Females to be interested in car insurance.

In [12]:
```
#analysis for Age
sns.FacetGrid(train, hue='Response', size = 5).map(sns.distplot, "Age").add_legend()
```

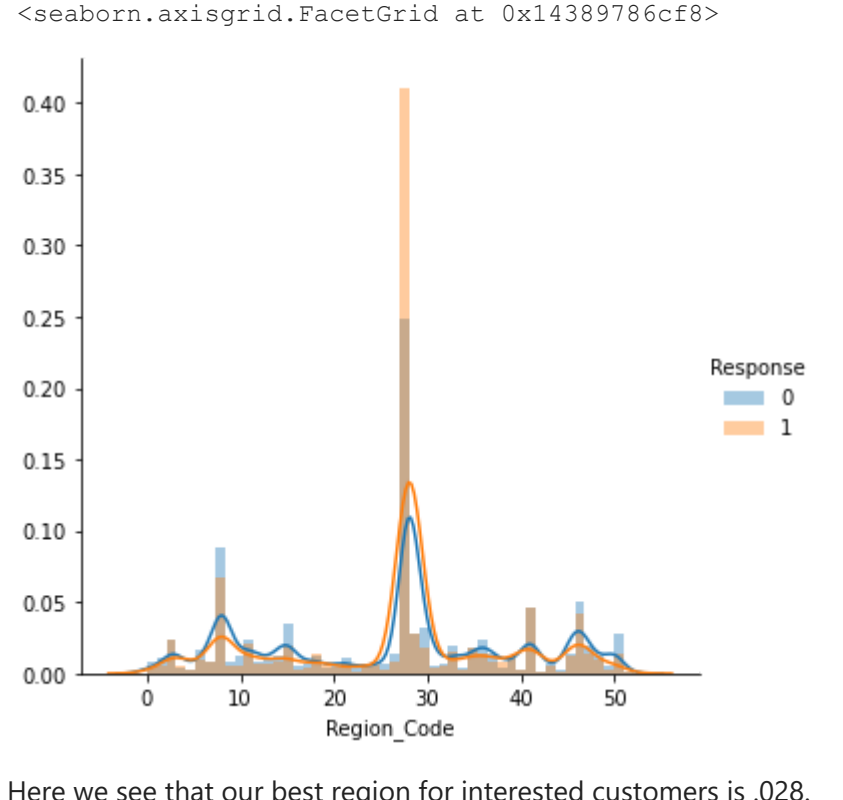
Out[12]: <seaborn.axisgrid.FacetGrid at 0x14384ae5ba8>



Here we see the sweet spot for interested customers is between ages 40 and 50. Furthermore, we see the sweet spot for customers not interested is between 20 and 30.

In [14]:
```
#analysis for Region Code
sns.FacetGrid(train, hue='Response', size = 5).map(sns.distplot, "Region_Code").add_legend()
```

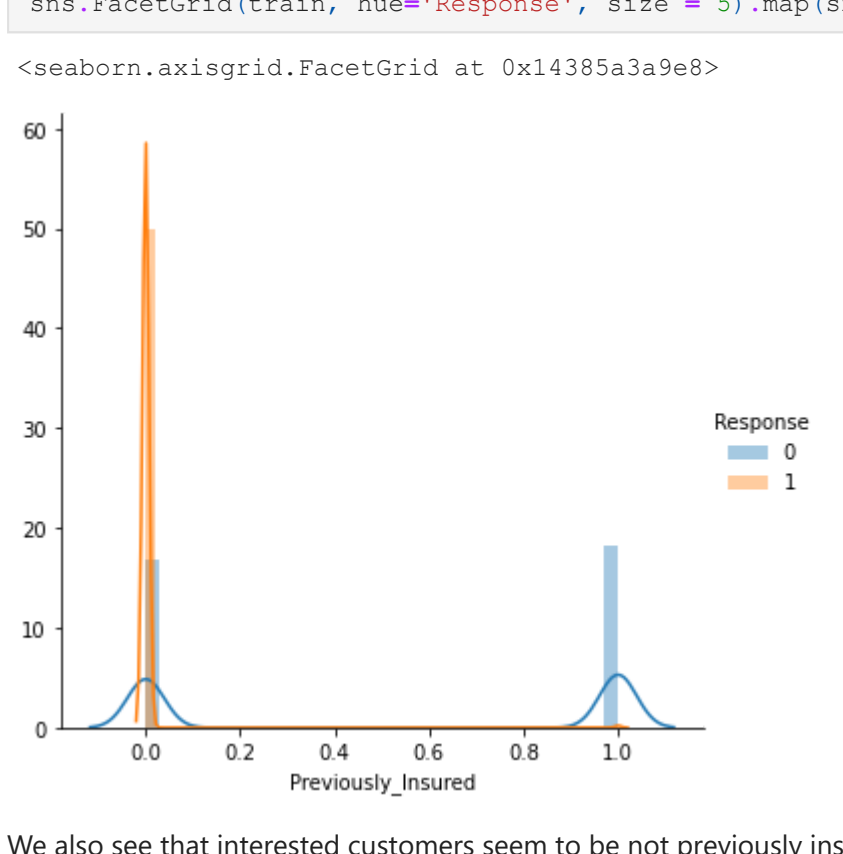
Out[14]: <seaborn.axisgrid.FacetGrid at 0x14389786cf8>



Here we see that our best region for interested customers is .028.

In [15]:
```
#analysis for Previously Insured
sns.FacetGrid(train, hue='Response', size = 5).map(sns.distplot, "Previously_Insured").add_legend()
```

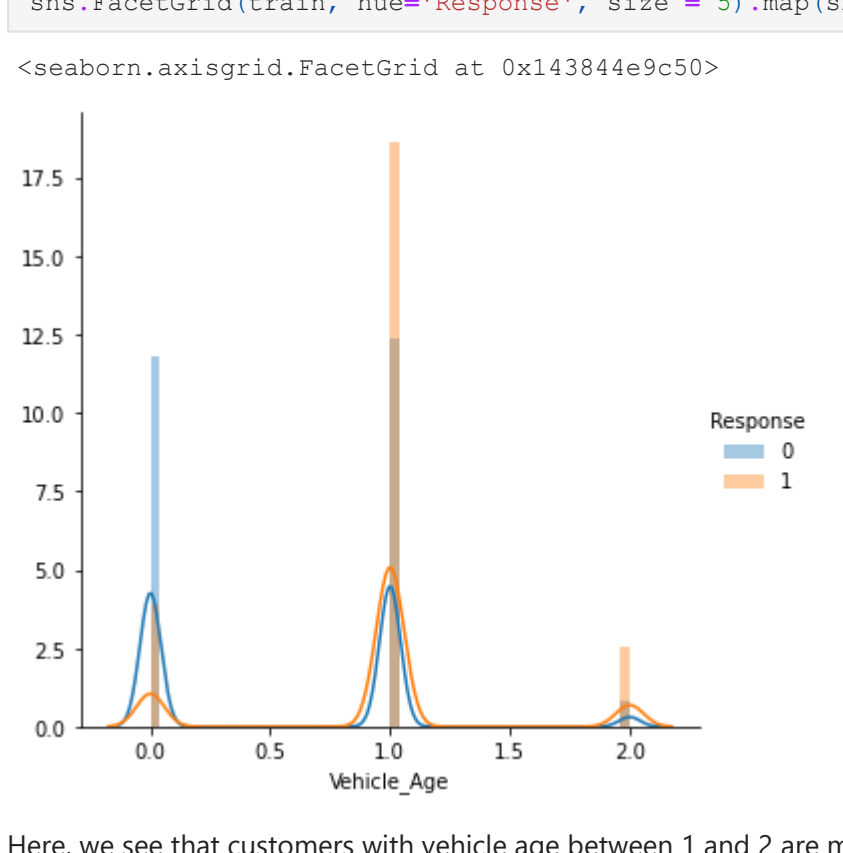
Out[15]: <seaborn.axisgrid.FacetGrid at 0x14385a3a9e8>



We also see that interested customers seem to be not previously insured.

In [27]:
```
#analysis for vehicle age
sns.FacetGrid(train, hue='Response', size = 5).map(sns.distplot, "Vehicle_Age").add_legend()
```

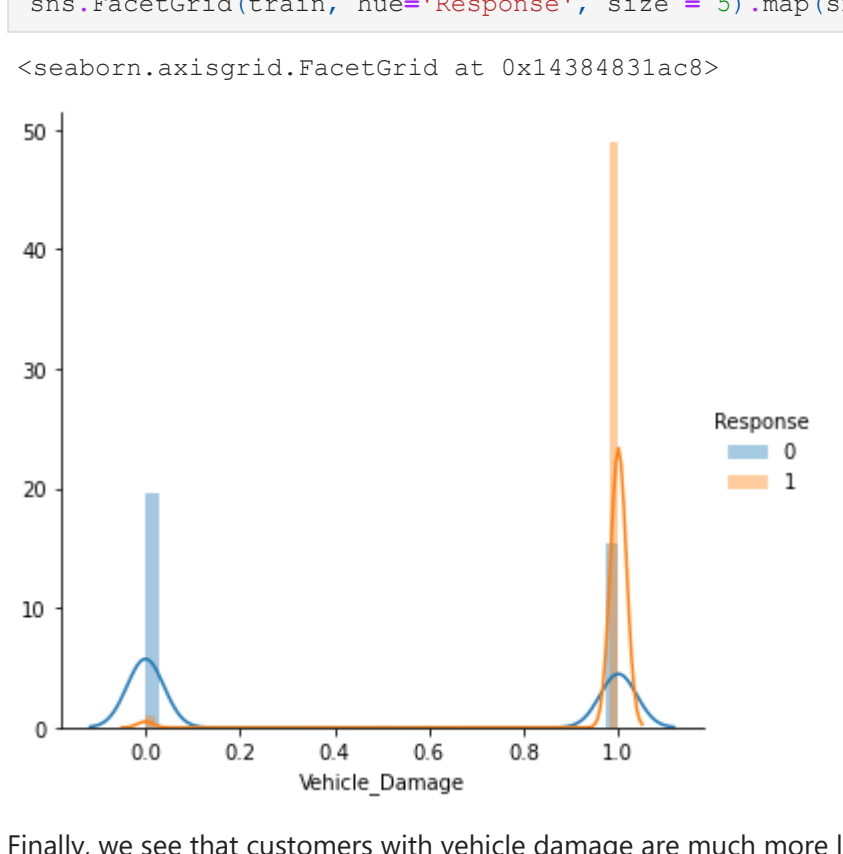
Out[27]: <seaborn.axisgrid.FacetGrid at 0x143844e9c50>



Here, we see that customers with vehicle age between 1 and 2 are much more likely to be interested.

In [28]:
```
#analysis for vehicle damage
sns.FacetGrid(train, hue='Response', size = 5).map(sns.distplot, "Vehicle_Damage").add_legend()
```

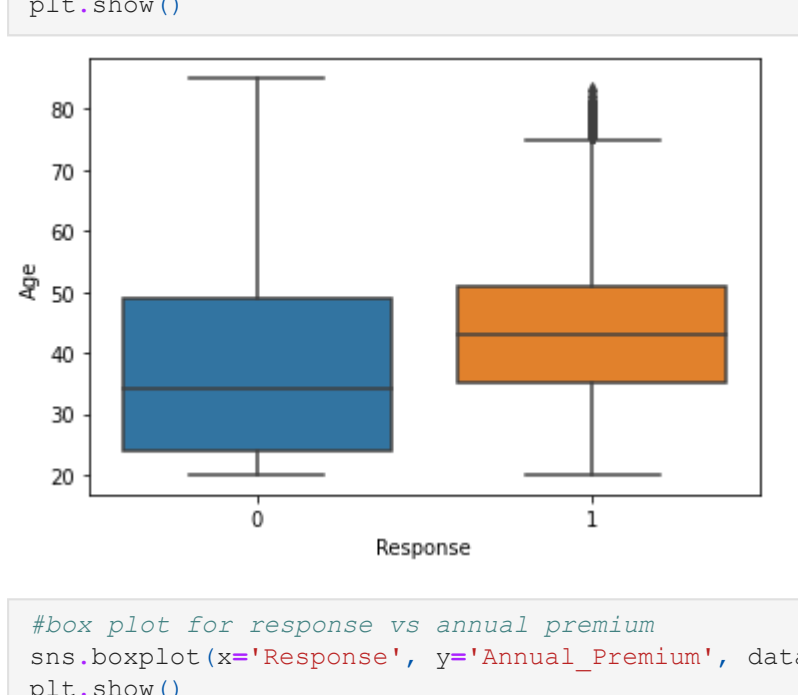
Out[28]: <seaborn.axisgrid.FacetGrid at 0x14384831ac8>



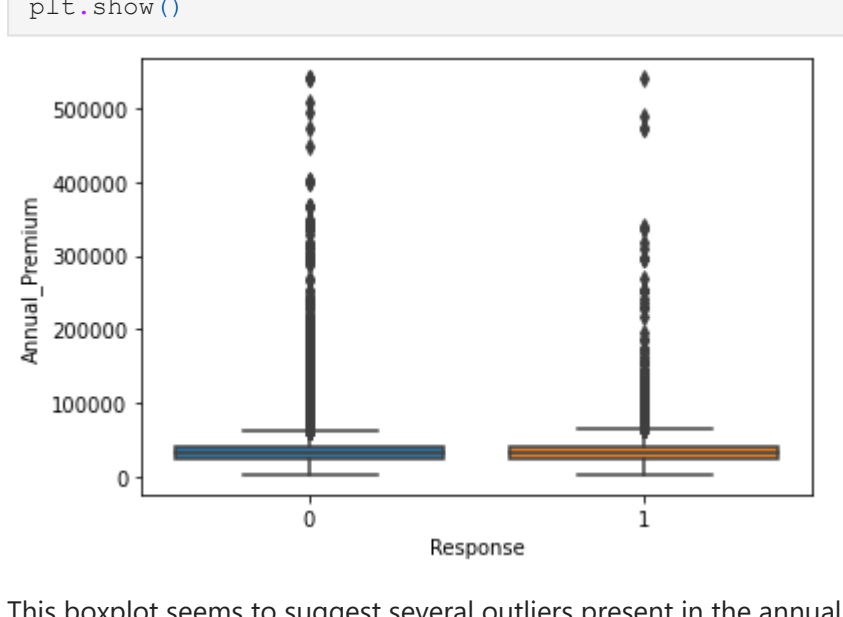
Finally, we see that customers with vehicle damage are much more likely to be interested in car insurance.

Bivariate Analysis

In [30]:
```
#box plot for response vs age
sns.boxplot(x='Response', y='Age', data=train)
plt.show()
```

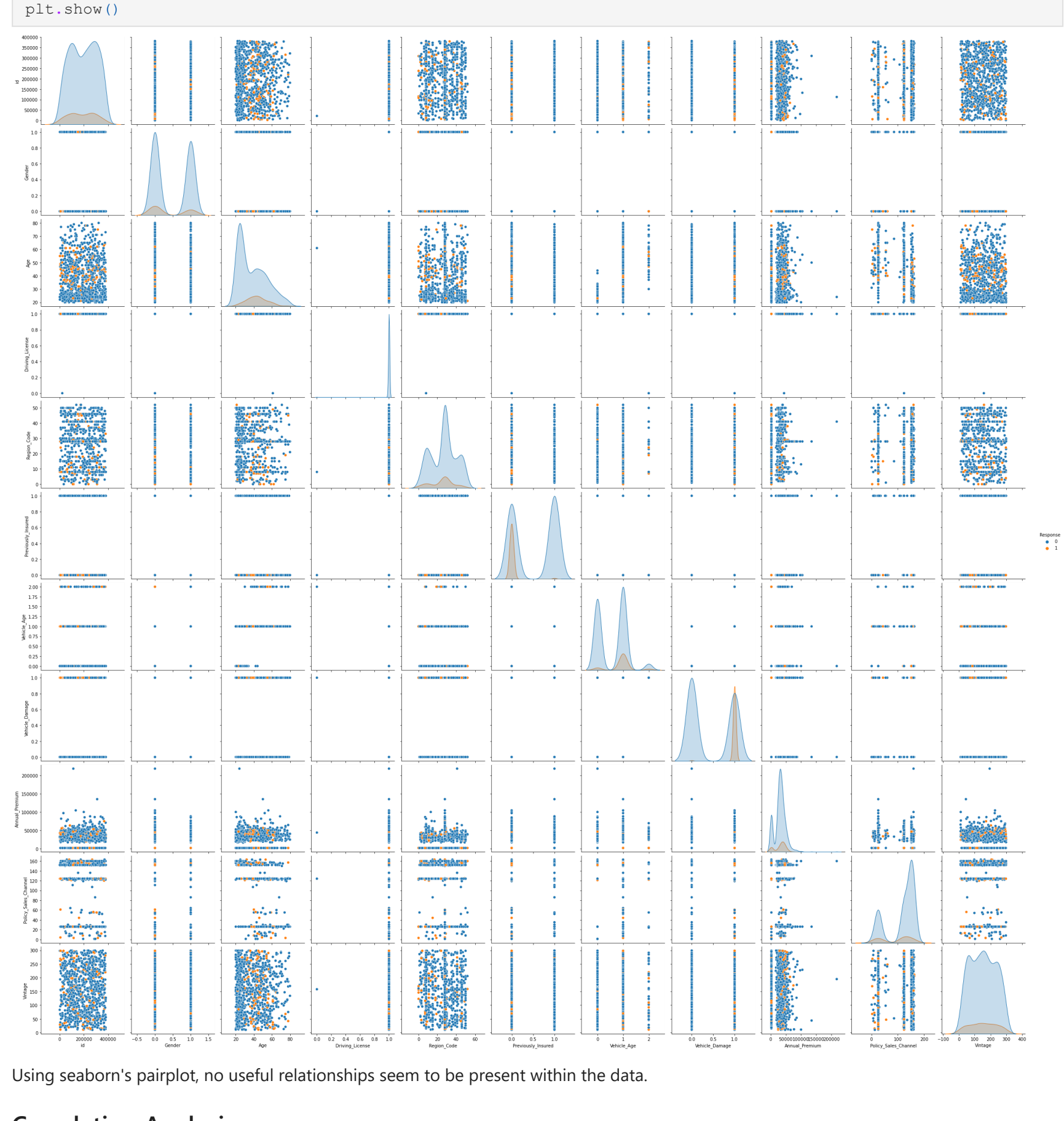


In [35]:
```
#box plot for response vs annual premium
sns.boxplot(x='Response', y='Annual_Premium', data=train)
plt.show()
```



This boxplot seems to suggest several outliers present in the annual_premium field. Further analysis on this later.

In [10]:
```
#pairplot for the data
sns.pairplot(train.sample(1000), hue='Response', size=3)
plt.show()
```

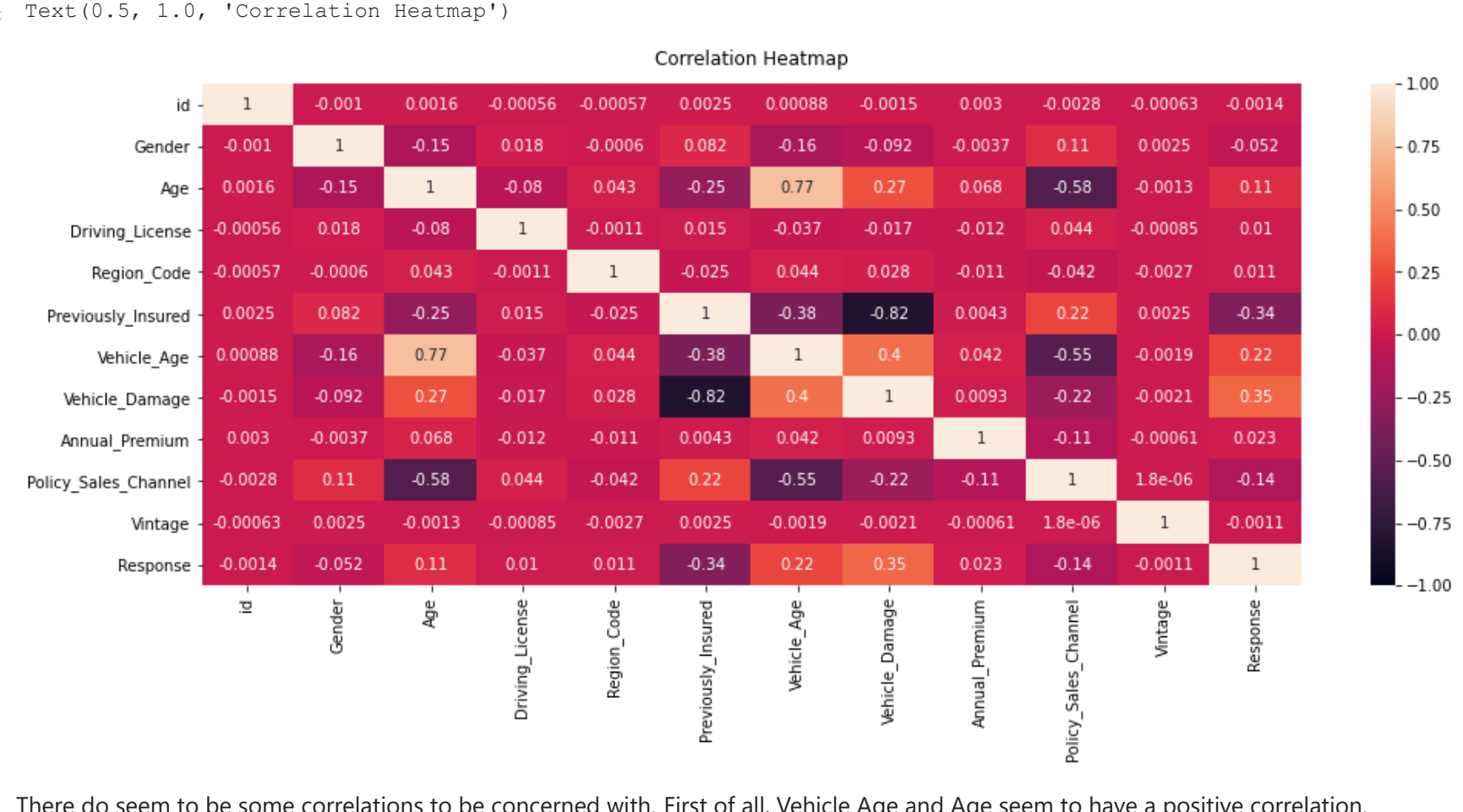


Using seaborn's pairplot, no useful relationships seem to be present within the data.

Correlation Analysis

In [11]:
```
#correlation plot to determine correlations between variables.
plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(train.corr(), vmin=-1, vmax=1, annot=True)
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12)
```

Out[11]: Text(0.5, 1.0, 'Correlation Heatmap')



There do seem to be some correlations to be concerned with. First of all, Vehicle Age and Age seem to have a positive correlation. Secondly, Previously Insured and Vehicle Damage also seem to have a negative correlation. We will leave this alone for now.

Outlier Detection

As we saw above, it appeared that annual premium had some outliers present. So we will perform additonal analysis on this column

In [6]:
```
#Calculate the IQR for Annual Premium
apIQR = train.Annual_Premium.quantile(.75) - train.Annual_Premium.quantile(.25)
```

In [7]:
```
#determine the rows with outliers and remove them
al_Premium'] >= train.Annual_Premium.quantile(.75) + 1.5*apIQR) |
```