VEHICLE INSURANCE CUSTOMER DATA

Contents :

---

1. Business Understanding

Context

Sebuah perusahaan vehicle insurance yang bertempat di wialayah Amerika Serikat bagian barat ingin memprediksi customer baru yang tertarik dengan produk asuransi mereka. Dimana calon customer tersebut di-approach dengan berbagai cara oleh perusahaan. Setelah banyak calon customer yang tertarik dan ingin membeli policy yang ditawarkan oleh tim sales & marketing, ternyata beberapa calon customer baru yang tidak relavan dengan preliminary yang telah dibuat oleh perusahaan. Melihat hal ini, tentunya akan merugikan perusahaan apabila calon customer tersebut tidak mampu untuk membayar premi per bulan kedepannya. Dari hal tersebut, maka perusahaan harus menyeleksi calon customer dengan response yes dan no kepada calon customer, apakah calon customer tersebut dianggap layak untuk membeli policy atau tidak. Oleh sebab itu perusahaan ingin mengetahui calon customer yang mana yang layak dan benar-benar mampu untuk membeli policy dan membayar premi per bulannya. Hal ini dilakukan untuk menjadi sebuah efektivitas dalam memilih calon customer dimana tentunya akan mengurangi waktu yang dibutuhkan dan biaya yang dikeluarkan oleh tim marketing. Di dalam database perusahaan ini, terdapat informasi mengenai demografi, pendidikan, income customer dll.

Target :

- **0 : Response no dari perusahaan (Customer yang dianggap tidak layak membeli policy)**
- **1 : Response yes dari perusahaan (Customer yang dianggap layak membeli policy)**

Problem Statement

Selama proses marketing, perusahaan menyadari bahwa sangat memakan waktu dan sumber daya jika perusahaan menargetkan semua calon customer tanpa melakukan penyaringan terlebih dahulu. Perusahaan ingin meningkatkan efisiensi marketing dengan mengetahui calon customer mana yang layak dan benar-benar mampu untuk membeli policy dan membayar premi per bulannya. Karena apabila tim sales & marketing meng-approach ke semua calon customer, maka waktu dan biaya marketing pun akan menjadi sia-sia jika berdasarkan kebijakan perusahaan calon customer yang tersebut tidak layak untuk membeli policy dan mendapatkan response no.

Goals

Dari permasalahan diatas, perusahaan ingin mencari tahu dan memprediksi kemungkinan calon customer mana yang layak dan benar-benar mampu untuk membeli policy dan membayar premi per bulannya dan mendapatkan response yes dari perusahaan. Sehingga perusahaan dapat fokus dan teknik marketing lebih efektif sehingga profit perusahaan menjadi naik.

Selain itu, perusahaan juga ingin mengetahui faktor apa saja yang bisa menjadi preliminary calon customer

Selain itu, perusahaan juga ingin mengetahui faktor apa saja yang bisa menjadi preliminary calon customer untuk membeli polis yang di tawarkan dan bagaimana perilaku dari calon customer yang ingin membeli polis. Sehingga perusahaan dapat membuat rencana yang lebih baik lagi dalam mendekati dan menyeleksi calon customer yang potensial untuk membeli.

Analytic Aproach

Dari isu yang telah digambarkan diatas, kita akan menganalisis data perusahaan yang telah ada sehingga kita dapat menemukan pola dan membedakan calon customer yang layak dan benar-benar mampu untuk membeli policy dan membayar premi per bulannya sehingga mendapatkan response yes dan no dari perusahaan.

Kemudian juga kita akan membuat machine learning dengan model klasifikasi yang akan membantu perusahaan untuk dapat memprediksi probabilitas seorang calon customer yang layak dan benar-benar mampu untuk membeli policy dan membayar premi per bulannya sehingga mendapatkan response yes dan no dari perusahaan.

Evaluation Metric

		PREDICTED	
		Response no dari perusahaan (Customer yang dianggap tidak layak membeli policy) (0)	Response yes dari perusahaan (Customer yang dianggap layak membeli policy) (1)
ACTUAL	Response no dari perusahaan (Customer yang dianggap tidak layak membeli policy) (0)	True Negative (TN) Model memprediksi calon nasabah tidak layak membeli polis, dengan aktual nasabah tidak layak membeli polis	False Positive (FP) Model memprediksi calon nasabah layak membeli polis, dengan aktual nasabah tidak layak membeli polis
	Response yes dari perusahaan (Customer yang dianggap layak membeli policy) (1)	False Negative (FN) Model memprediksi calon nasabah tidak layak membeli polis, dengan aktual nasabah layak membeli polis	True Positive (TP) Model memprediksi calon nasabah layak membeli polis, dengan aktual nasabah layak membeli polis

- Type 1 error : False Positive
  Konsekuensi : Perusahaan berpotensi kehilangan nasabah dikarenakan calon customer sulit untuk membayar sehingga perusahaan akan kehilangan waktu, sumber daya dan biaya approach calon customer
- Type 2 error : False Negative
  Konsekuensi : Perusahaan kehilangan calon customer yang potensial

Berdasarkan konsekuensi diatas, maka seharusnya yang kita lakukan adalah membuat model yang dapat mengurangi false negative sehingga perusahaan kekurangan (kehilangan) calon customer yang potensial. Hal ini dikarenakan jika terjadi Type 1 error, perusahaan dapat mencabut hak claim calon customer yang tidak dapat membayar premi (menunggak). Namun juka terjadi Type 2 error, maka perusahaan akan sangat merugi dikarenakan kehilangan 100% pendapatan dari calon customer yang potensial.

Oleh karena itu yang harus kita lakukan adalah membuat model dengan tipe :

- Accuracy : Rasio prediksi Benar (positif dan negatif) dengan keseluruhan data (berapa persen calon customer yang benar diprediksi mendapatkan response yes dan tidak mendapatkan response yes dari keseluruhan calon customer?)
- Recall : Rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif (berapa persen calon customer yang diprediksi mendapatkan response yes dari keseluruhan calon customer yang sebenarnya mendapatkan response yes?)

Dari hal tersebut, metric utama yang akan kita gunakan adalah `Accuracy` dan `Recall`.

---

2. Data Understanding

Data Source : Vehicle Insurance Customer Data

- Dataset diambil dari salah satu perusahaan Vehicle Insurance di Amerika Serikat tahun 2011
- Sebagian besar fitur bersifat kategori (Nominal, Ordinary, Binary), dengan beberapa fitur mempunyai kardinalitas yang tinggi

- **Terdapat 24 kolom, dimana setiap baris dari kolom tersebut merepresentasikan informasi seorang kandidat yang mempunyai tanggal efektif asuransi di tahun 2011**
- **Target model adalah variabel response**

Attribute Information

Attribute	Data Type	Description
Customer	object	Customer's unique ID
State	object	State of customer
Customer Lifetime Value	float64	Indicator value of customer by company
Response	object	Company's response to Customer's claim
Coverage	object	Coverage of Policy
Education	object	Customer's education
Effective To Date	object	Last effective date of policy
EmploymentStatus	object	Customer's status employment
Gender	object	Customer's gender
Income	int64	Customer's income
Location Code	object	Code of location area
Marital Status	object	Customer's marital status
Monthly Premium Auto	int64	Monthly policy debit
Months Since Last Claim	int64	Months since customer last claim
Months Since Policy Inception	int64	Months since customer registered/issued
Number of Open Complaints	int64	Number of open complaints by customer
Number of Policies	int64	Number of open policies by customer
Policy Type	object	A type of policy that customer claimed
Policy	object	A category of policy type that customer claimed
Renew Offer Type	object	Type of renewal offer
Sales Channel	object	Type of approach customer by company
Total Claim Amount	float64	An amount of customer claimed
Vehicle Class	Object	A type of class vehicle that insuranced
Vehicle Size	Object	A type of size vehicle that insuranced

In [1]:

```
# Import library for exploring dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns


# To ignores the warnings
import warnings
warnings.filterwarnings('ignore')
```

Read the dataset

In [2]:

```
# Read the dataset
df = pd.read_csv('AutoInsurance.csv')
```

```
df.head()
```

Out[2]:

	Customer	State	Customer Lifetime Value	Response	Coverage	Education	Effective To Date	EmploymentStatus	Gender	Income	...
0	BU79786	Washington	2763.519279	No	Basic	Bachelor	2/24/11	Employed	F	56274	...
1	QZ44356	Arizona	6979.535903	No	Extended	Bachelor	1/31/11	Unemployed	F	0	...
2	AI49188	Nevada	12887.431650	No	Premium	Bachelor	2/19/11	Employed	F	48767	...
3	WW63253	California	7645.861827	No	Basic	Bachelor	1/20/11	Unemployed	M	0	...
4	HB64268	Washington	2813.692575	No	Basic	Bachelor	3/2/2011	Employed	M	43836	...

5 rows × 24 columns

In [3]:

```
# Check the dataFeatures, dataType, null, nullPct, unique and uniqueSample each feature
listItem = []
for col in df.columns :
    listItem.append([col, df[col].dtype, df[col].isna().sum(), round((df[col].isna().sum()/len(df[col])) * 100,2),
                     df[col].nunique(), list(df[col].drop_duplicates().sample(2).values)]);

dfDesc = pd.DataFrame(columns=['dataFeatures', 'dataType', 'null', 'nullPct', 'unique', 'uniqueSample'],
                      data=listItem)
dfDesc
```

Out[3]:

	dataFeatures	dataType	null	nullPct	unique	uniqueSample
0	Customer	object	0	0.0	9134	[WC21120, WH76688]
1	State	object	0	0.0	5	[California, Oregon]
2	Customer Lifetime Value	float64	0	0.0	8041	[5169.558551, 4001.519098]
3	Response	object	0	0.0	2	[No, Yes]
4	Coverage	object	0	0.0	3	[Basic, Extended]
5	Education	object	0	0.0	5	[High School or Below, College]
6	Effective To Date	object	0	0.0	59	[8/2/2011, 2/14/11]
7	EmploymentStatus	object	0	0.0	5	[Retired, Medical Leave]
8	Gender	object	0	0.0	2	[M, F]
9	Income	int64	0	0.0	5694	[27951, 65709]
10	Location Code	object	0	0.0	3	[Suburban, Rural]
11	Marital Status	object	0	0.0	3	[Divorced, Married]
12	Monthly Premium Auto	int64	0	0.0	202	[242, 167]
13	Months Since Last Claim	int64	0	0.0	36	[21, 32]
14	Months Since Policy Inception	int64	0	0.0	100	[34, 41]

	dataFeatures	dataType	null	nullPct	unique	uniqueSample
15	Number of Open Complaints	int64	0	0.0	6	[4, 1]
16	Number of Policies	int64	0	0.0	9	[2, 7]
17	Policy Type	object	0	0.0	3	[Personal Auto, Special Auto]
18	Policy	object	0	0.0	9	[Corporate L3, Corporate L1]
19	Renew Offer Type	object	0	0.0	4	[Offer2, Offer1]
20	Sales Channel	object	0	0.0	4	[Agent, Web]
21	Total Claim Amount	float64	0	0.0	5106	[127.104785, 56.692777]
22	Vehicle Class	object	0	0.0	6	[Two-Door Car, Luxury Car]
23	Vehicle Size	object	0	0.0	3	[Large, Small]

Dari info yang ditampilkan diatas terlihat bahwa dataset ini memiliki 24 kolom dengan 9134 baris data dan tidak memiliki missing-value di dalamnya. Dimana setiap kolomnya memiliki tipe data yang berbeda-beda. Dapat juga kita pastikan bahwa dataset ini juga tidak memiliki duplicated data. Hal ini dikarenakan jumlah unique data pada kolom `Customer` sama dengan jumlah baris data yang ada.

In [4]:

```
# Recheck for duplicated rows
print("Number of duplicated rows: ", sum(df.duplicated()))
```

```
Number of duplicated rows:  0
```

3. Exploratory Data Analysis

Sebelum melakukan Exploratory Data Analysis, pertama kita akan men-drop variable `Customer` karena kolom ini hanya mengandung id unik yg membedakan setiap kandidat dan tidak berguna untuk analisa dan pembuatan model machine learning nantinya.

In [5]:

```
# Drop column Customer
df.drop(['Customer'],axis=1, inplace=True)
```

In [6]:

```
# Describe numerical feature on dataset
df.describe()
```

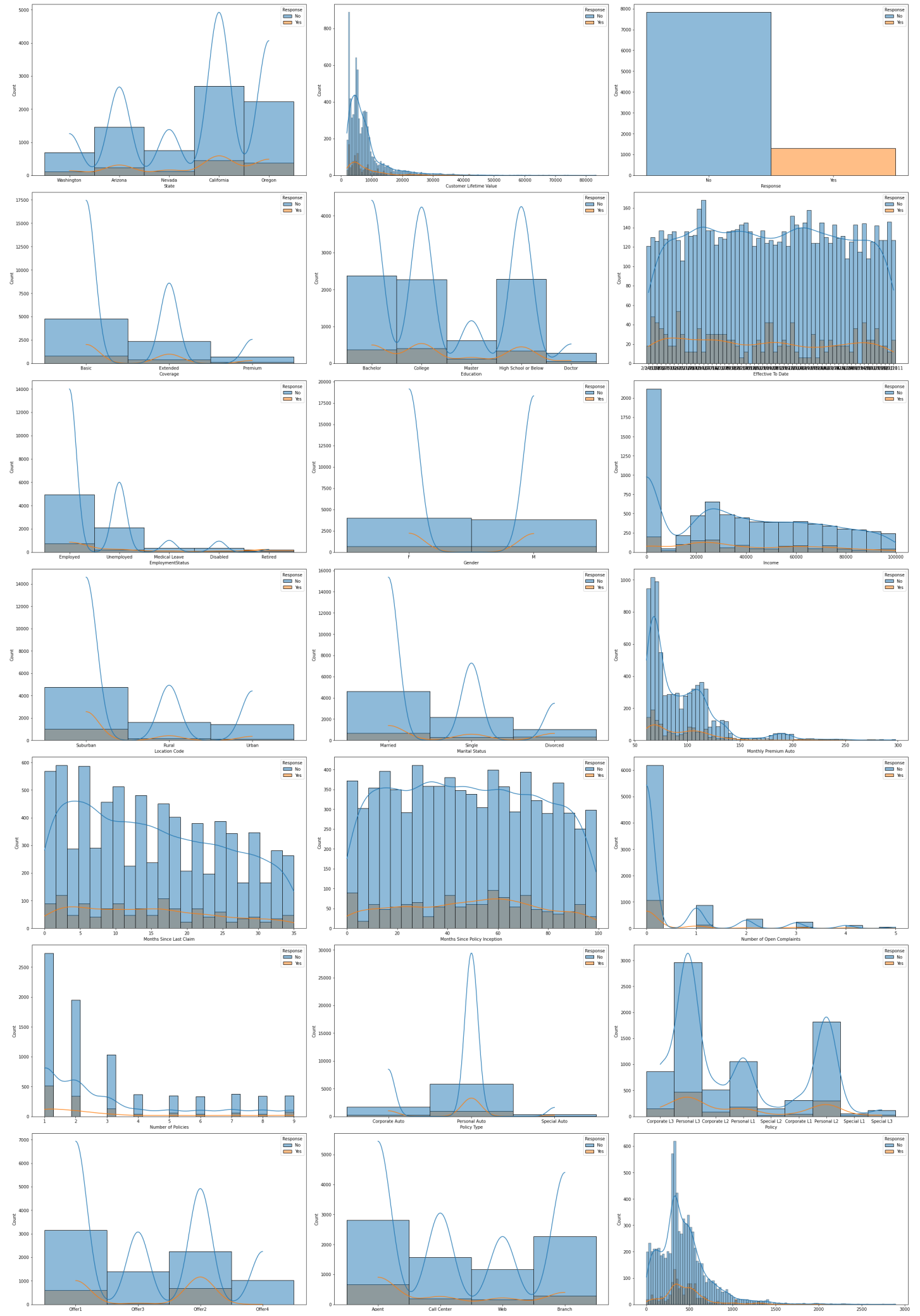
Out[6]:

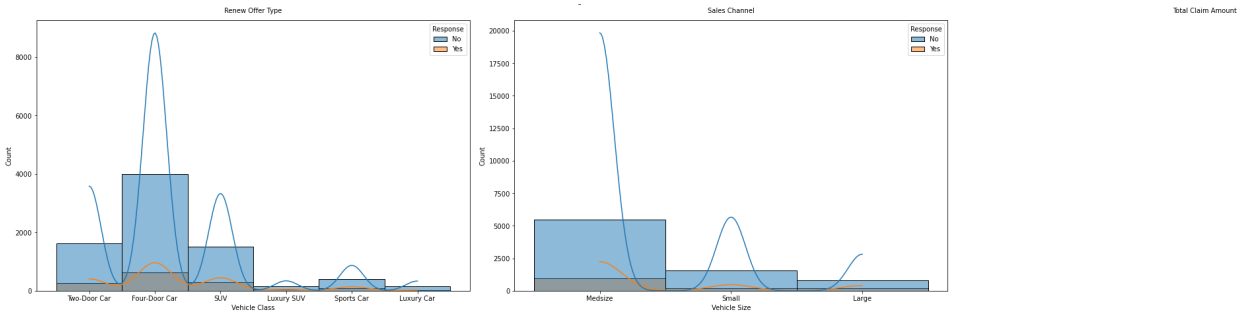
	Customer Lifetime Value	Income	Monthly Premium Auto	Months Since Last Claim	Months Since Policy Inception	Number of Open Complaints	Number of Policies	Total Claim Amount
count	9134.000000	9134.000000	9134.000000	9134.000000	9134.000000	9134.000000	9134.000000	9134.000000
mean	8004.940475	37657.380009	93.219291	15.097000	48.064594	0.384388	2.966170	434.088794
std	6870.967608	30379.904734	34.407967	10.073257	27.905991	0.910384	2.390182	290.500092
min	1898.007675	0.000000	61.000000	0.000000	0.000000	0.000000	1.000000	0.099007
25%	3994.251794	0.000000	68.000000	6.000000	24.000000	0.000000	1.000000	272.258244
50%	5780.182197	33889.500000	83.000000	14.000000	48.000000	0.000000	2.000000	383.945434
75%	8962.167041	62320.000000	109.000000	23.000000	71.000000	0.000000	4.000000	547.514839
max	83325.381190	99981.000000	298.000000	35.000000	99.000000	5.000000	9.000000	2893.239678

In [7]:

```
# Graph each variable with response
```

```
fig = plt.figure(figsize=(30, 50), constrained_layout=True)
for i in range(len(df.columns)):
    plt.subplot(8, 3, i+1)
    sns.histplot(data=df, x=df[df.columns[i]], hue='Response', kde=True)
```





Key Points :

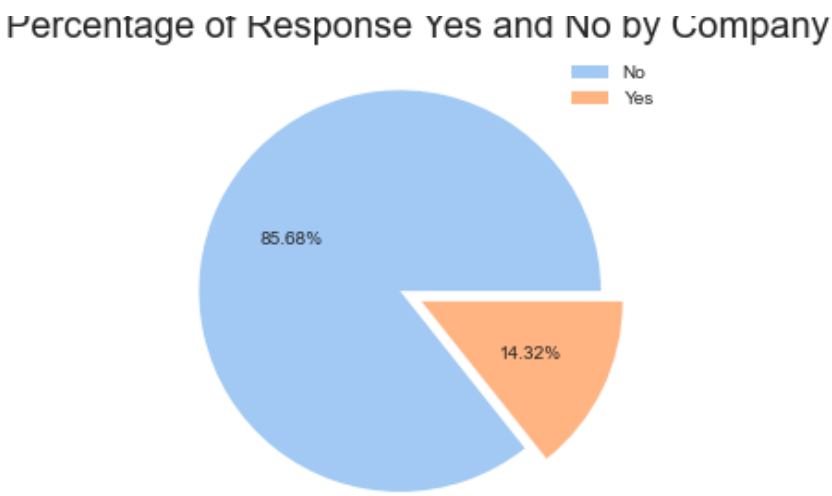
- **Negara bagian California dan Oregon merupakan negara bagian dengan calon customer tertinggi dan juga memiliki calon dengan response yes tertinggi dari perusahaan**
- **Banyak calon customer dengan CLV (Customer Lifetime Value) kurang dari 10000 namun tetap memiliki response yes yang tinggi oleh perusahaan**
- **Lebih dari 70% calon customer mendapat response no dari perusahaan**
- **Calon customer lebih banyak tertarik untuk memilih basic coverage dan extended coverage untuk policy mereka**
- **Rata-rata calon customer memiliki jenjang pendidikan bachelor, college dan highschool or below**
- **Lebih dari 50% calon customer memiliki pekerjaan**
- **Berdasarkan frekuensinya (jumlah dan respon yes), gender dari calon customer hampir sama rata baik itu laki-laki maupun perempuan**
- **Secara income, ternyata banyak calon customer memiliki income kurang dari 60000 U.S. Dollar**
- **Lokasi area calon customer sebagian besar berasal dari Suburban (pinggiran kota)**
- **Melihat dari status pernikahan, banyak calon customer yang sudah menikah**
- **Banyak calon customer yang memilih polisi dengan premi (pembayaran debit) kurang dari 150 U.S. Dollar per bulan**
- **Sebagian besar calon customer tidak pernah membuat complain**
- **Banyak calon customer yang hanya memiliki satu sampai dengan tiga polis. Dimana masih sedikit calon customer yang memiliki empat sampai tujuh polis.**
- **Banyak calon customer yang mendaftar polis dengan tipe personal dengan tipe polis personal L3 merupakan tipe polis terfavorit**
- **Calon Customer dengan tipe offer 1 dan tipe offer 2 memiliki respon yes terbanyak secara keseluruhan**
- **Marketing dengan cara Agent & Branch lebih banyak mendapatkan calon customer daripada teknik marketing lainnya**
- **Secara total claim, banyak calon customer yang mengajukan claim sampai dengan 1000 U.S. Dollar, namun terdapat calon customer yang memiliki total claim lebih dari itu**
- **Kebanyakan mobil yang diasuransikan adalah mobil dengan empat pintu (hampir 50% dari jumlah total customer)**
- **Berkaitan dengan fitur vehicle class, mobil dengan ukuran medium size merupakan mobil dengan ukuran yang paling banyak di asuransi kan oleh customer.**

Setelah mendapatkan key points dari subplot diatas, kita olah kembali key points tersebut menjadi bivariat dan multivariat analisis guna untuk melihat dan menganalisis dataset lebih dalam. Sehingga kita mendapatkan insight dan gambaran pola lebih jelas dari calon customer.

Dive-in Persentage of Response Yes and No by Company

In [8]:

```
# Dive-in Persentage of Response Yes and No by Company
plt.style.use('seaborn')
plt.figure(figsize=(10,5))
colors = sns.color_palette('pastel')
plt.pie(df['Response'].value_counts().values, explode= (0.1,0.02), colors = colors,
        autopct='%.2f%%')
plt.title ('Percentage of Response Yes and No by Company', size = 20)
plt.legend(df['Response'].unique())
plt.show()
```



Berdasarkan semua calon customer yang ada, jika di persentasikan company hanya memberikan sekitar 14.32% response yes terhadap semua calon customer yang mendaftar. Dilirik dari sisi bisnis, kemungkinan banyak variable-variable yang menjadi pertimbangan company untuk menerima applicant dari calon customer mereka.

Hal ini juga kita sadari bahwa pada target data kita nantinya (response) memiliki imbalance data. Hal ini perlu kita note dan proses pada tahap selanjutnya (Preprocessing).

Define dataframe Response Yes

Menilik evaluation metric yang telah dijelaskan. Dimana nantinya kita akan menggunakan metric `Accuracy` dan `Recall` yang berfokus kepada prediksi benar response yes, maka kita harus membuat terlebih dahulu data frame yang hanya berisikan value response yes untuk aggregating analisa selanjutnya.

In [9]:

```
# Define df Yes for Aggregating Percentage of Yes Response
dfy = df[(df.Response == 'Yes')]
```

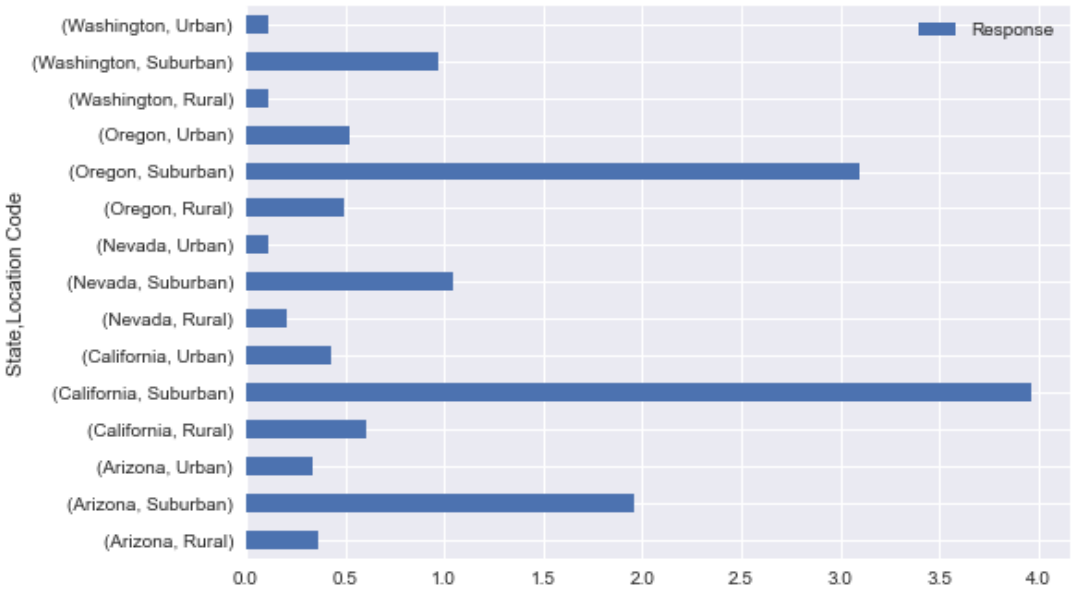
Count Response of State and Location Code by Yes Response as Percentage

In [13]:

```
# Count Response of State and Location Code by Yes Response as Percentage
(dfy.groupby(['State','Location Code']).count()[['Response']]/df.groupby(['State','Location Code']).count()[['Response']].sum()*100).plot.barh()
```

Out[13]:

```
<AxesSubplot:ylabel='State,Location Code'>
```



Terlihat bahwa di setiap negara bagian, wilayah pinggiran kota (suburban area) merupakan wilayah dengan calon customer terbanyak mendapatkan response yes dari perusahaan secara persentasi. Hal ini kemungkinan besar disebabkan oleh mayoritas calon customer di wilayah ini bekerja di wilayah kota. Sehingga dengan selisih penghasilan yang lebih tinggi dan biaya hidup yang lebih rendah, sehingga perusahaan berasumsi jika calon customer berkemungkinan besar dapat membayar premi per bulannya.

Hal yang menarik lainnya adalah perusahaan ternyata lebih condong untuk memberikan response yes ke calon customer yang berada di wilayah pedesaan (rural area) dibandingkan dengan calon customer yang berada di wilayah kota (urban area). Hal ini dikarenakan kecelakaan mobil di Amerika Serikat secara umum banyak terjadi di wilayah perkotaan daripada wilayah perdesaan dan pinggiran kota. Menilik hal ini, perusahaan tentunya tidak ingin mengeluarkan banyak uang dan rugi untuk mengcover polis calon customer yang sering complain. *Sumber 1*, *Sumber 2*

In [11]:

```
# Crosstab Response of Number of Policies and Number of Open Complaints by Yes Response a
s Percentage
pd.crosstab(dfy['Number of Open Complaints'],dfy['Number of Policies'])
```

Out[11]:

Number of Policies	1	2	3	4	5	6	7	8	9
Number of Open Complaints									
0	426	300	102	30	42	30	48	30	54
1	42	24	24	12	6	6	12	6	6
2	12	6	6	0	0	0	0	0	0
3	18	12	0	0	6	0	0	6	12
4	18	0	0	0	0	6	0	0	0
5	0	0	0	0	6	0	0	0	0

Dari crosstab diatas dapat dilihat bahwa sebanyak apapun polis yang dimiliki oleh calon customer dan jika calon customer semakin sedikit mangajukan complaints, maka response yes dari perusahaan lebih banyak.

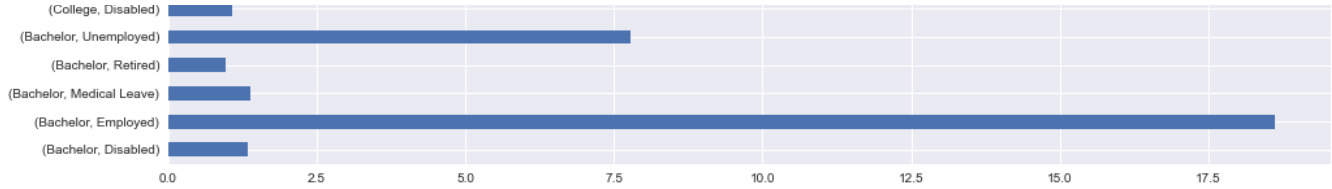
Graph Response of Education and EmploymentStatus Code as Percentage

In [12]:

```
# Graph Response of Education and EmploymentStatus Code as Percentage
(df.groupby(['Education','EmploymentStatus']).count()[['Response']]/df.groupby(['Educati
on','EmploymentStatus']).count()[['Response']].sum()*100).plot.barh(legend=True, figsize
=(16,10))
```

Out[12]:

```
<AxesSubplot:ylabel='Education,EmploymentStatus'>
```



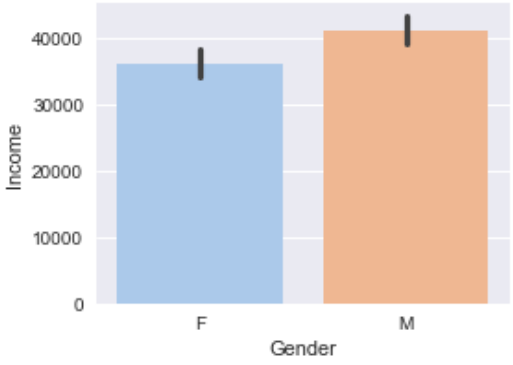
Dari grafik diatas dapat kita lihat bahwa perusahaan banyak memberikan response kepada calon customer mempunyai pekerjaan (Employed). Namun kita juga dapat melihat bahwa perusahaan tetap memberikan response kepada calon customer yang tidak mempunyai pekerjaan (Unemployed). Dengan demikian, walaupun kemungkinan besar calon customer yang bekerja dapat membayar premi per bulannya, namun status pekerjaan calon customer bukan menjadi salah satu preliminary oleh perusahaan untuk memberikan asuransinya. *Sumber 3*, *Sumber 4*

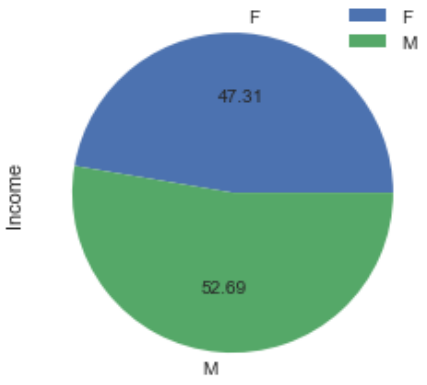
Namun, ketika kita melihat lebih jauh, ternyata perusahaan memberikan response yang lebih sedikit kepada calon customer yang ber status pensiun (retired). Melihat dari analisa dan sumber sebelumnya, kemungkinan besar hal ini terjadi karena masa pensiun seseorang berkorelasi dengan umur sesorang. Secara umum, orang yang pensiun (retired) merupakan orang-orang yang sudah berumur. Dimana semakin tua seseorang, maka akan semakin berbahaya ketika terjadi kecelakaan. Perusahaan tentunya akan mengurangi response dari orang yang sudah berumur dikarenakan banyakanya tanggungan yang harus diberikan oleh perusahaan apabila terjadi kecelakaan. Dari analisa tersebut dapat disimpulkan bahwa umur dari seorang calon customer bisa menjadi salah satu preliminary oleh perusahaan untuk memberikan asuransinya. *Sumber 5*

Count Response of Gender, Marital Status and Income by Yes Response as Percentage

In [13]:

```
# Income by Customer's Gender by Yes Response as Percentage
plt.style.use('seaborn')
plt.figure(figsize=(4,3))
sns.barplot(x='Gender', y='Income', data=dfy, palette="pastel")

GenderIncomePercentage = dfy.groupby(['Gender']).sum()[['Income']]
GenderIncomePercentage.plot.pie(autopct="%.2f",figsize=(7,4), subplots=True)
plt.show()
```

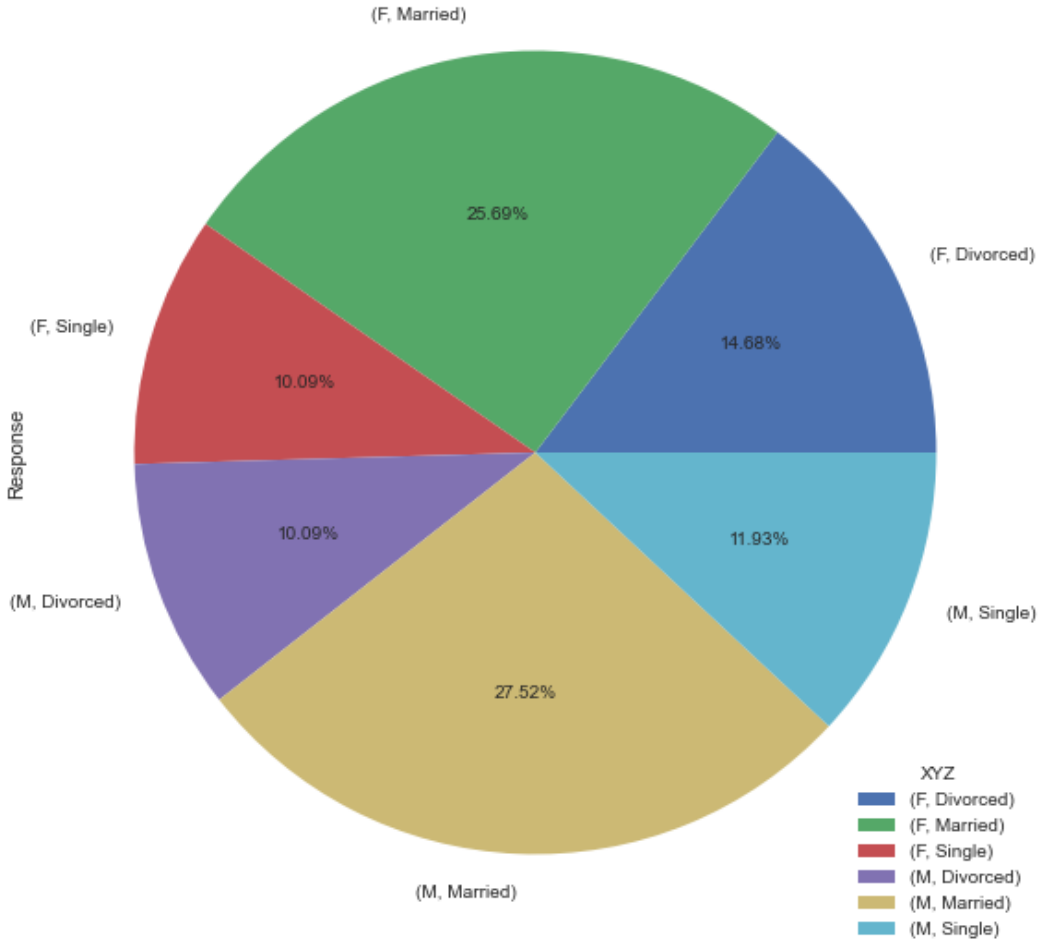




In [14]:

```
# Count Response of Gender and Marital Status by Yes Response as Percentage
GaMS = (dfy.groupby(['Gender','Marital Status']).count()[['Response']]/df.groupby(['Gend
```

```
er','Marital Status']).count()[['Response']].sum()*100)
GaMS.plot.pie(subplots=True, autopct='%.2f%%', figsize=(15,10))
plt.legend(title='XYZ')
plt.show()
```



Dari grafik sebelumnya kita mengetahui bahwa berdasarkan frekuensinya (jumlah dan respon yes), gender dari calon customer hampir sama rata baik itu laki-laki maupun perempuan. Namun setelah kita hubungkan dengan variable marital status & income, terlihat bahwa sedikit ada perbedaan diantaranya.

Dari hubungan variabel diatas, secara persentase perusahaan lebih memberikan response yes kepada calon customer dengan semua gender yang sudah menikah. Hal ini kemungkinan disebabkan karena calon customer dianggap lebih financially stable dengan menggabungkan gaji mereka (joint income), sehingga customer jenis ini merupakan customer yang sangat aman dimiliki oleh perusahaan. Di dalam perusahaan auto/car insurance juga biasanya memembuat produk polis paket bundling seperti multi-driver atau multi-car agar customer mendapatkan discount. Oleh karena itu, calon customer dengan yang sudah menikah pastinya akan lebih tertarik dengan paket yang diberikan dan perusahaan akan mendapatkan keuntungan dari sektor penjualan paket bundling tersebut.

Berdasarkan status single dan divorced, agregasi response yes dari perusahaan ternyata berbeda pada tiap gender. Calon customer laki-laki lebih banyak mendapatkan response yes ketika berstatus single dibandingkan dengan calon customer perempuan. Namun lain halnya ketika kedua calon customer tersebut berstatus divorced, perusahaan memberikan reponse yes lebih banyak kepada calon customer perempuan daripada laki-laki. Hal-hal tersebut dimungkinkan oleh secara income, calon customer laki-laki lebih besar daripada perempuan sehingga calon customer laki-laki lebih banyak mendapatkan response yes ketika berstatus single. Namun ketika terjadi perceraian, secara umum risk yang didapatkan oleh laki-laki lebih besar tanggungannya daripada perempuan dan kemungkinan akan terjadi calon customer laki-laki harus membayar premi 'bersama' menjadi tanggungan pribadi. Hal ini tentunya akan memberatkan calon customer itu sendiri dan perusahaan tentunya tidak mau calon customer menunggak iuran premi. *Sumber 6*, *Sumber 7*

Crosstab Response of Sales Channel and Policy Type by Yes Response as Percentage

In [15]:

```
# Crosstab Response of Sales Channel and Policy Type by Yes Response as Percentage
pd.crosstab(dfy['Sales Channel'],dfy['Policy Type'])/df['Response'].count()*100
```

Out[15]:

Policy Type / Sales Channel	Corporate Auto	Personal Auto	Special Auto
Agent	1.467046	5.419312	0.405080
Branch	0.810160	2.342895	0.065689
Call Center	0.481717	1.521787	0.098533
Web	0.394132	1.204292	0.109481

Secara rata-rata dari data dengan response yes, ternyata calon customer lebih banyak memilih tipe personal daripada tipe polis lainnya, sedangkan Special Auto menjadi tipe polis yang paling sedikit terjual. Hal ini dapat kita simpulkan bahwa ternyata banyak mobil yang diasuransikan merupakan mobil milik pribadi calon customer. Dimana agregat dari response yes yang company berikan pun cukup terlihat signifikan diantara ketiga tipe polis tersebut.

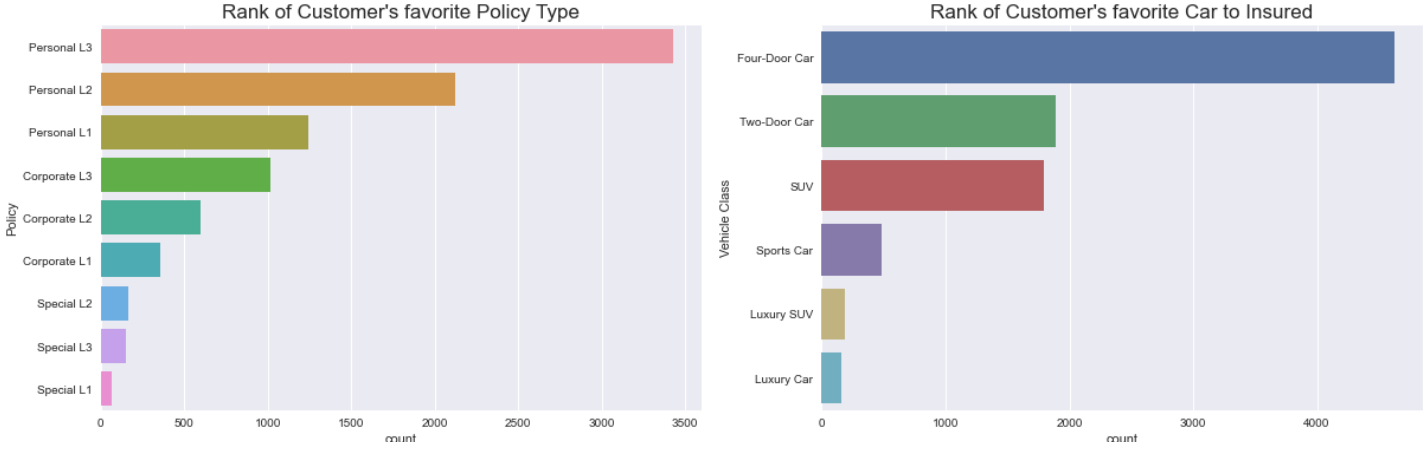
Mengapa produk polis dengan tipe Special Auto memiliki persentase yang paling sedikit diantara penjualan tipe polis lainnya? Kemungkinan besar hal ini terjadi karena tipe special auto ini merupakan tipe polis dengan memberikan bantuan penerbitan polis oleh perusahaan kepada calon customer yang dianggap tidak mampu membayar. Tentunya jika perusahaan memberikan response yes yang lebih banyak daripada tipe polis lainnya, polis ini pun akan banyak terjual akan sangat riskan terhadap kelangsungan bisnis perusahaan. *Sumber 8*

Selain itu, jika dilihat dari data penjualan, sales channel tipe agent memiliki agregasi response yes yang paling banyak menjual dari keseluruhan sales channel (marketing channel). Melihat dari cara marketing agent dengan approach ke calon customer secara B2C (business to customer), adanya komisi atas hasil setiap penjualan yang dilakukan (sehingga memberikan motivasi tersendiri) dan mengetahui apa produk yang sesuai dengan calon customer, tidak heran bahwa sales channel dengan tipe agent yang paling banyak menjual polis dengan response yes oleh perusahaan disusul dengan Branch, Call Center dan Web.

Rank Product by Customer Favorite and Yes Response as Percentage

In [16]:

```
# Rank Product by Customer Favorite
plt.figure(figsize=(20,6))

plt.subplot(1,2,1)
sns.countplot(y=df['Policy'], order = df['Policy'].value_counts().index)
plt.title("Rank of Customer's favorite Policy Type", size = 17)
plt.xticks(size = 10)

plt.subplot(1,2,2)
sns.countplot(y=df['Vehicle Class'], order = df['Vehicle Class'].value_counts().index)
plt.title("Rank of Customer's favorite Car to Insured", size = 17)
plt.xticks(size = 10)

plt.show()
```



In [17]:

```
# Crosstab Response of Policy Type and Vehicle Size by Yes Response as Percentage
pd.crosstab(df['Policy Type'],df['Vehicle Size'])/df['Response'].count()*100
```

Out[17]:

Vehicle Size	Large	Medsize	Small
Policy Type			
Corporate Auto	2.167725	15.283556	4.094592
Personal Auto	7.795051	52.178673	14.342019
Special Auto	0.394132	2.868404	0.875848

Melihat dari grafik diatas, semua insurance tipe personal (L3, L2 & L1) merupakan top three product favorite customer, dengan mobil terbanyak di asuransikan bertipe Four-Door Car, Two-Door Car dan SUV. Melihat dari *Sumber 9* mobil yang mempunyai vehicle size medsize adalah mobil yang mempunyai karakteristik 130–159 cubic feet (3,680 – 4,500 liter). Secara umum kebanyakan mobil Four-Door Car, SUV dan Two-Door Car masuk didalam kategori ini.

Jika kita melihat berdasarkan rata-rata, mobil bertipe Medsize merupakan mobil yang di asuransikan terbanyak oleh customer. Dimana Tipe mobil medsize ini paling banyak diasuransikan dengan polis tipe personal auto. Artinya kebanyakan mobil yang diasuransikan merupakan mobil yang berukuran medium (Four-Door Car, SUV dan Two-Door Car) dan merupakan mobil pribadi calon customer.

In [18]:

```
# Crosstab Response of Vehicle Size and Vehicle Class by Yes Response as Percentage
pd.crosstab(dfy['Vehicle Size'],dfy['Vehicle Class'])/df['Response'].count()*100
```

Out[18]:

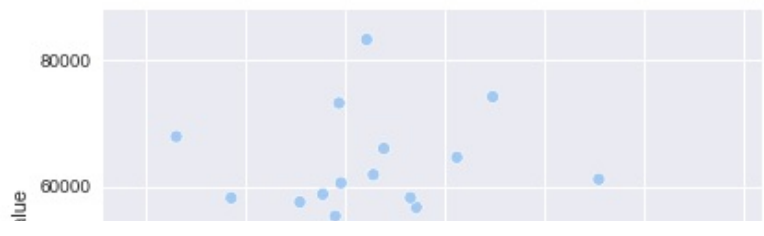
Vehicle Class	Four-Door Car	Luxury Car	Luxury SUV	SUV	Sports Car	Two-Door Car
Vehicle Size						
Large	0.919641	0.000000	0.000000	0.394132	0.131377	0.394132
Medsize	4.795270	0.065689	0.262755	2.364791	0.853952	1.970659
Small	1.116707	0.065689	0.065689	0.394132	0.000000	0.525509

Secara bisnis, perusahaan lebih memilih mobil Medsize dari setiap lini vehicle class untuk di response yes. Hal ini terjadi karena jika melihat dari persebaran data sebelumnya (Rank of Customer's favorite Car to Insured) dan juga range harga sparepart mobil pada tipe ini secara umum lebih murah daripada (Luxury Car, Luxury SUV dan Sports Car) tentunya laba yang didapatkan oleh perusahaan akan lebih jika banyak menjual polis dan berfokus marketingnya di lini mobil Medsize.

Mobil-mobil mahal tentunya mempunyai range harga sparepart yang mahal juga. Belum lagi apabila barang tersebut indent, claimed dari calon customer pun akan lebih mahal dan akan sangat merugikan perusahaan apabila banyak meresponse yes pada lini mobil tipe ini. *Sumber 10*

In [19]:

```
# Correlation Customer Value and Total Claim Amount
sns.pairplot(data=df, y_vars='Customer Lifetime Value', x_vars='Total Claim Amount', hue=
'Response', height=5, aspect=1.2, palette="pastel").map(sns.kdeplot, levels=1);
```



Melihat dari korelasi antara Total Claim Amount dan Customer Lifetime Value dari calon customer, secara rata-rata banyak calon customer yang hanya mengclaim kurang dari 1500 dollar secara total. Kita juga dapat melihat adanya korelasi disini. Dimana juga terlihat bahwa perusahaan memberikan skor CLV (Customer Lifetime Value) yang cukup besar untuk calon customer yang melakukan total claim lebih rendah. Begitu pula sebaliknya, perusahaan akan memberikan skor CLV yang rendah untuk calon customer dengan total claim yang besar. Hal ini dilakukan oleh perusahaan agar perusahaan tidak rugi dan lebih aware terhadap covering calon customer nantinya.

Namun tidak dipungkiri bahwa perusahaan akan menilik variable lain sebagai acuan nilai CLV ini. Oleh sebab itu pada beberapa poin, perusahaan memberikan response yes ketika nilai CLV nasabah diluar dari rata-rata. Kemungkinan terbesar hal ini dilihat bahwa calon customer memiliki riwayat pembayaran yang bagus pada transaksi sebelumnya. *Sumber 11*

In [20]:

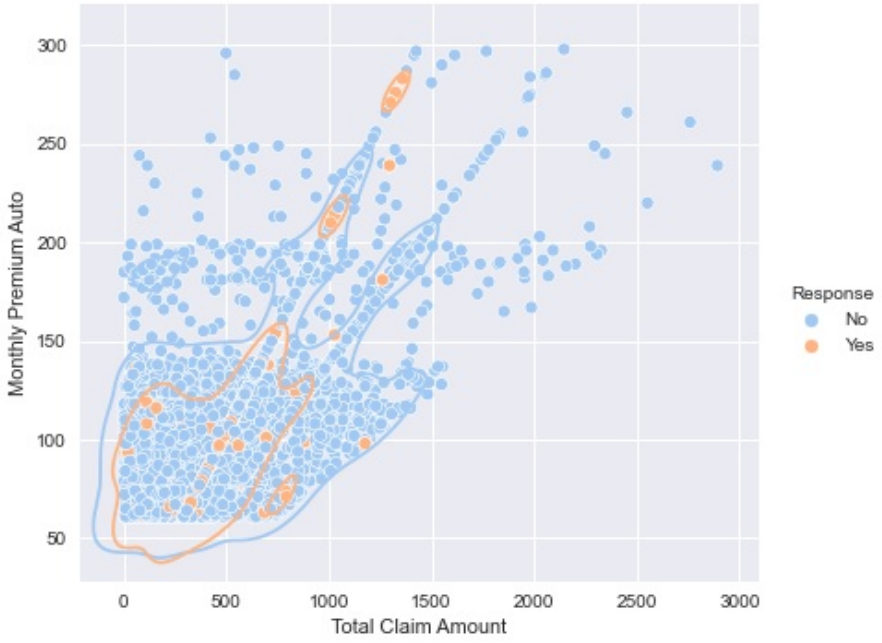
```
# Pairplot Customer Value and Monthly Premium Auto
sns.pairplot(data=df, y_vars='Customer Lifetime Value', x_vars='Monthly Premium Auto', hu
e='Response', height=5, aspect=1.2, palette="pastel").map(sns.kdeplot, levels=1);
```

Selain dari variable Total Claim Amount, ternyata perusahaan juga menilai calon customer berdasarkan variable Monthly Premium Auto. Dimana terlihat pola yang cukup jelas diantara keduanya. Secara umum perusahaan akan memberikan skor CLV yang lebih tinggi untuk calon customer dengan Monthly Premium Auto yang lebih besar. Tentunya perusahaan akan menilai calon customer yang membeli polis murah 'kurang berharga' untuk bisnis dibandingkan dengan calon customer yang membeli polis mahal.

In [21]:

```
# Pairplot Multivariat Analysis ; Monthly Premium Auto, Total Claim Amount and Policy Typ
e
sns.pairplot(data=df, y_vars='Monthly Premium Auto', x_vars='Total Claim Amount', hue='Re
```

```
sponse', height=5, aspect=1.2, palette="pastel").map(sns.kdeplot, levels=1);
```



Melihat dari pairplot sebelumnya, ternyata memang ada korelasi antara variable Monthly Premium Auto dan Total Claim Amount. Pada beberapa keadaan, semakin besar Monthly Premium Auto yang diambil oleh calon customer maka akan semakin besar pula Total Claim yang dilakukan oleh calon customer. Tentunya calon customer tidak ingin menyia-nyiakan polis yang telah dibeli.

4. Preprocessing

Dari info sebelumnya dijelaskan bahwa dataset yang kita miliki tidak memiliki missing-value di dalamnya. Dimana setiap kolomnya memiliki tipe data yang berbeda-beda. Dapat juga kita pastikan bahwa dataset ini juga tidak memiliki duplicated data karena jumlah unique data pada kolom `Customer` sama dengan jumlah baris data yang ada. Dan kita juga sudah membuang kolom `Customer` pada proses sebelumnya karena hanya berisi data unique id dari masing-masing customer. Maka pada Preprocessing kali ini kita hanya akan melakukan Data Cleaning, Data Transform dan Preview Outlier dari dataset yang dimiliki.

Data Cleaning

In [22]:

```
# Check the dataFeatures, dataType, null, nullPct, unique and uniqueSample each feature
listItem = []
for col in df.columns :
    listItem.append([col, df[col].dtype, df[col].isna().sum(), round((df[col].isna().sum
()/len(df[col])) * 100,2),
                     df[col].nunique(), list(df[col].drop_duplicates().sample(2).values)]
);

dfDesc = pd.DataFrame(columns=['dataFeatures', 'dataType', 'null', 'nullPct', 'unique',
'uniqueSample'],
                      data=listItem)
dfDesc
```

Out[22]:

	dataFeatures	dataType	null	nullPct	unique	uniqueSample
0	State	object	0	0.0	5	[Washington, Nevada]
1	Customer Lifetime Value	float64	0	0.0	8041	[15488.43196, 2300.334605]
2	Response	object	0	0.0	2	[Yes, No]
3	Coverage	object	0	0.0	3	[Extended, Basic]

	dataFeatures	dataType	null	nullPct	unique	uniqueSample
4	Education	object	0	0.0	5	[Doctor, Master]
5	Effective To Date	object	0	0.0	59	[1/18/11, 3/1/2011]
6	EmploymentStatus	object	0	0.0	5	[Unemployed, Disabled]
7	Gender	object	0	0.0	2	[F, M]
8	Income	int64	0	0.0	5694	[75037, 25572]
9	Location Code	object	0	0.0	3	[Rural, Suburban]
10	Marital Status	object	0	0.0	3	[Married, Divorced]
11	Monthly Premium Auto	int64	0	0.0	202	[232, 159]
12	Months Since Last Claim	int64	0	0.0	36	[19, 28]
13	Months Since Policy Inception	int64	0	0.0	100	[26, 42]
14	Number of Open Complaints	int64	0	0.0	6	[2, 0]
15	Number of Policies	int64	0	0.0	9	[8, 5]
16	Policy Type	object	0	0.0	3	[Corporate Auto, Personal Auto]
17	Policy	object	0	0.0	9	[Corporate L3, Personal L3]
18	Renew Offer Type	object	0	0.0	4	[Offer4, Offer2]
19	Sales Channel	object	0	0.0	4	[Agent, Web]
20	Total Claim Amount	float64	0	0.0	5106	[501.994635, 65.803328]
21	Vehicle Class	object	0	0.0	6	[Luxury Car, Luxury SUV]
22	Vehicle Size	object	0	0.0	3	[Large, Small]

Terlihat bahwa kolom `Customer` sudah tidak ada di dalam dataset.

Berikutnya kita lihat kolom `Effective To Date`. Dapat kita lihat bahwa pada kolom ini tipe datanya object, dimana seharusnya kolom ini mempunyai tipe data datetype. Kita juga dapat melihat pada uniqueSample bahwa cara penanggalan pada kolom ini tidak beraturan. Dengan jumlah uniqueSample sebanyak 59 buah dan isi kolom hanya menjelaskan tentang tanggal efektif berakhir insurance nasabah di tahun 2011, maka kita akan men-drop kolom `Effective To Date` dari dataset kita.

In [23]:

```
# Drop column Effective To Date
df.drop(['Effective To Date'],axis=1, inplace=True)
```

In [24]:

```
# Preview Cleaned Data
listItem = []
for col in df.columns :
    listItem.append([col, df[col].dtype, df[col].isna().sum(), round((df[col].isna().sum()/len(df[col])) * 100,2),
                     df[col].nunique(), list(df[col].drop_duplicates().sample(2).values)]);

dfDesc = pd.DataFrame(columns=['dataFeatures', 'dataType', 'null', 'nullPct', 'unique', 'uniqueSample'],
                      data=listItem)
dfDesc
```

Out[24]:

	dataFeatures	dataType	null	nullPct	unique	uniqueSample
0	State	object	0	0.0	5	[Arizona, Nevada]
1	Customer Lifetime Value	float64	0	0.0	8041	[5640.729767, 6612.220871]
2	Response	object	0	0.0	2	[Yes, No]
3	Coverage	object	0	0.0	3	[Basic, Premium]

	dataFeatures	dataType	null	nullPct	unique	uniqueSample
4	Education	object	0	0.0	5	[Doctor, College]
5	EmploymentStatus	object	0	0.0	5	[Employed, Unemployed]
6	Gender	object	0	0.0	2	[M, F]
7	Income	int64	0	0.0	5694	[25918, 20009]
8	Location Code	object	0	0.0	3	[Urban, Rural]
9	Marital Status	object	0	0.0	3	[Divorced, Single]
10	Monthly Premium Auto	int64	0	0.0	202	[193, 63]
11	Months Since Last Claim	int64	0	0.0	36	[7, 22]
12	Months Since Policy Inception	int64	0	0.0	100	[85, 10]
13	Number of Open Complaints	int64	0	0.0	6	[3, 1]
14	Number of Policies	int64	0	0.0	9	[7, 3]
15	Policy Type	object	0	0.0	3	[Corporate Auto, Personal Auto]
16	Policy	object	0	0.0	9	[Personal L3, Corporate L2]
17	Renew Offer Type	object	0	0.0	4	[Offer1, Offer2]
18	Sales Channel	object	0	0.0	4	[Agent, Call Center]
19	Total Claim Amount	float64	0	0.0	5106	[45.425684, 22.819088]
20	Vehicle Class	object	0	0.0	6	[Four-Door Car, Two-Door Car]
21	Vehicle Size	object	0	0.0	3	[Large, Medsize]

Preview Outliers

Sebelum melihat dan menganalisa outlier dari dataset, pertama kita akan mengimport `stat` dari module `scipy` dan setelah itu mendefinisikan fungsi Numerical Variable dari data(df) untuk memudahkan analisa outliernya nanti.

In [25]:

```
# Import module for stats
from scipy import stats
from scipy.stats import shapiro

# Define the Numerical_Variable
Numerical_Variable = df[['Customer Lifetime Value', 'Income', 'Monthly Premium Auto', 'Months Since Last Claim', 'Months Since Policy Inception','Number of Open Complaints','Number of Policies','Total Claim Amount']]
```

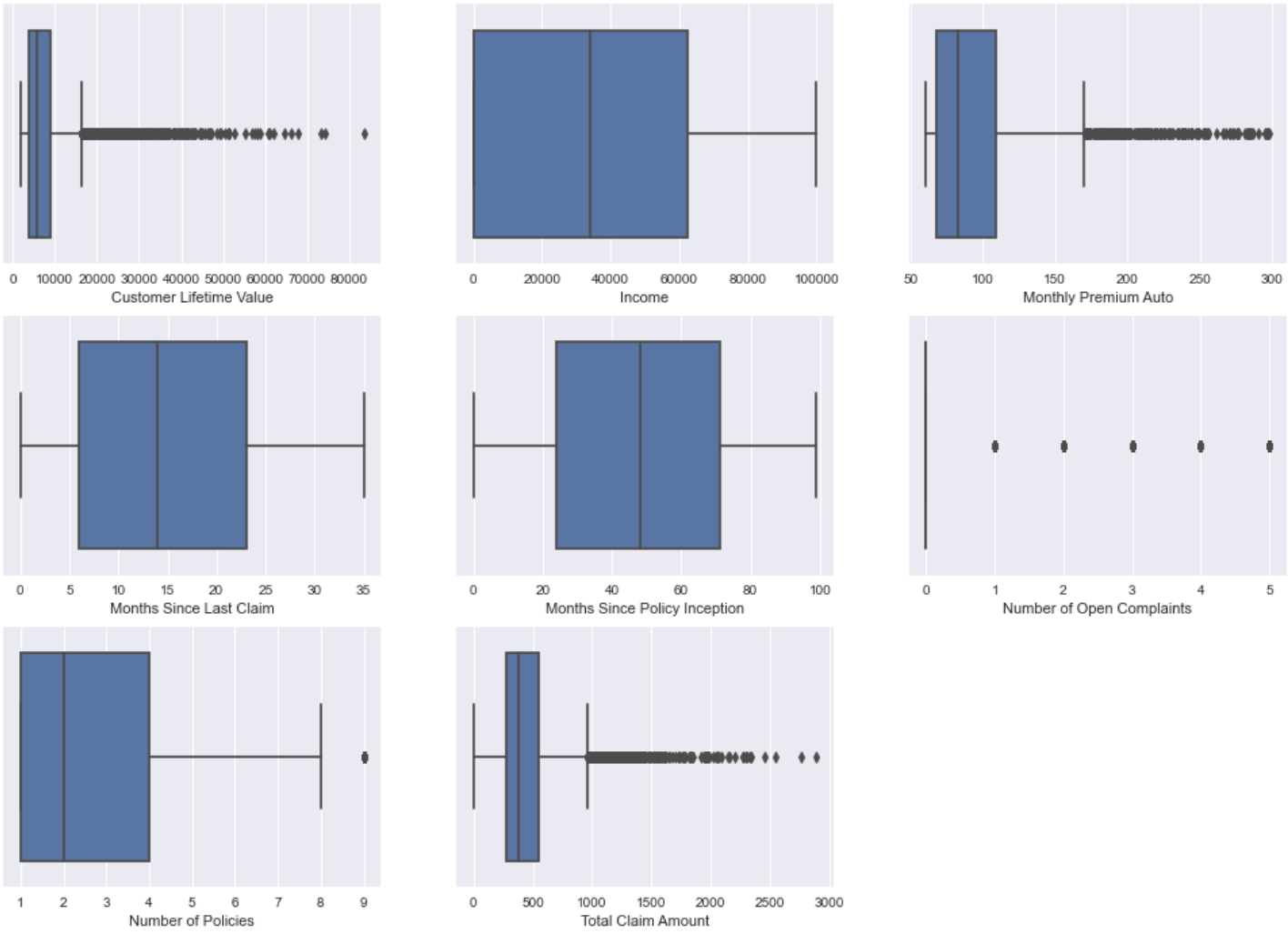
In [26]:

```
plt.figure(figsize=(17,12))

plt.subplot(331)
sns.boxplot(data=Numerical_Variable,x='Customer Lifetime Value')

plt.subplot(332)
sns.boxplot(data=Numerical_Variable,x='Income')

plt.subplot(333)
sns.boxplot(data=Numerical_Variable,x='Monthly Premium Auto')

plt.subplot(334)
sns.boxplot(data=Numerical_Variable,x='Months Since Last Claim')

plt.subplot(335)
sns.boxplot(data=Numerical_Variable,x='Months Since Policy Inception')

plt.subplot(336)
sns.boxplot(data=Numerical_Variable,x='Number of Open Complaints')
```

```
plt.subplot(337)
sns.boxplot(data=Numerical_Variable,x='Number of Policies')

plt.subplot(338)
sns.boxplot(data=Numerical_Variable,x='Total Claim Amount')
```

Out[26]:

```
<AxesSubplot:xlabel='Total Claim Amount'>
```

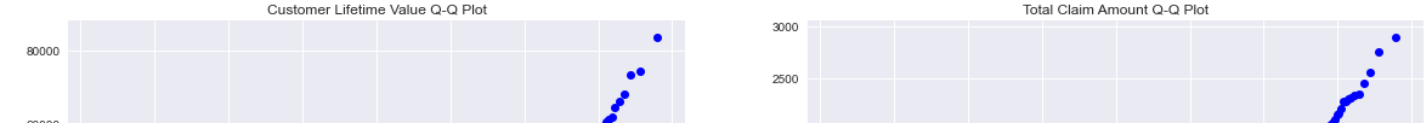


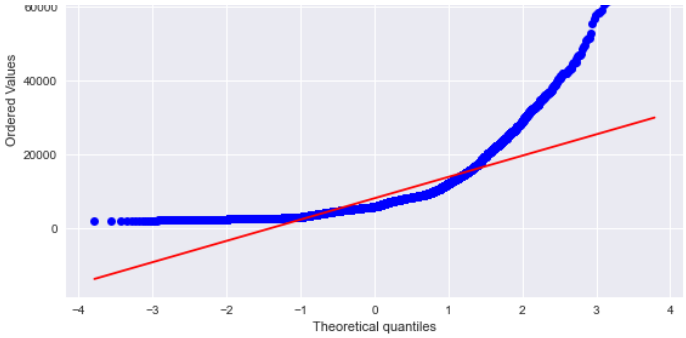
Dari boxplot diatas, kita dapat melihat bahwa data kita hanya sedikit memiliki outlier. Dimana outlier terbanyak terdapat di variable `Customer Lifetime Value` dan beberapa di variable `Total Claim Amount`. Melihat distribusi datanya, nilai outlier yang terdapat pada kedua variable tersebut tidak terlalu ekstrim.

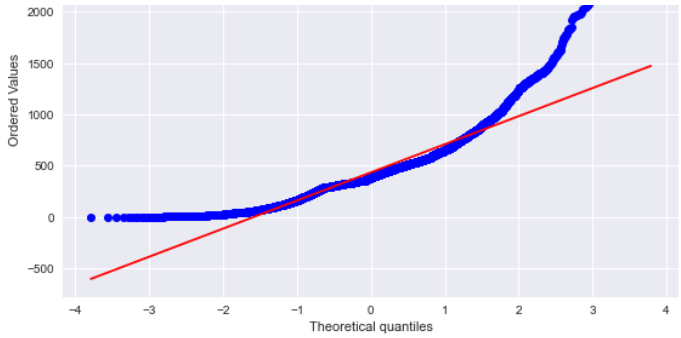
Mari kita q-q plotting data dari kedua variable ini. ***Sumber 12***

In [27]:

```
# Rank product by customer favorite
plt.figure(figsize=(20,6))

plt.subplot(1,2,1)
stats.probplot(Numerical_Variable['Customer Lifetime Value'], dist="norm", plot=plt)
plt.title("Customer Lifetime Value Q-Q Plot")
plt.savefig("Customer_Lifetime_Value_Q-Q_Plot.png")

plt.subplot(1,2,2)
stats.probplot(Numerical_Variable['Total Claim Amount'], dist="norm", plot=plt)
plt.title("Total Claim Amount Q-Q Plot")
plt.savefig("Total_Claim_Amount_QQ.png")

plt.show()
```







Dari kedua grafik diatas dapat dilihat bahwa adanya penyimpangan normalitas terutama pada awal dan akhir data. Pada kedua grafik diatas terlihat penyimpangan terjadi pada nilai di bawah -1 dan nilai diatas 1.5.

Mari kita uji kedua variable tersebut secara statistik untuk melihat apakah data terdistribusi normal. Untuk menguji ini, kita dapat menggunakan uji Saphiro-Wilk untuk uji normalitas. *Sumber 13*

In [28]:

```
# Shapiro-Wilk test
print('Saphiro-Wilk Customer Lifetime Value :',stats.shapiro(Numerical_Variable['Customer Lifetime Value']))
print('Saphiro-Wilk Total Claim Amount :',stats.shapiro(Numerical_Variable['Total Claim Amount']))
```

```
Saphiro-Wilk Customer Lifetime Value : ShapiroResult(statistic=0.7033728361129761, pvalue=0.0)
Saphiro-Wilk Total Claim Amount : ShapiroResult(statistic=0.8883205056190491, pvalue=0.0)
```

Melihat hasil dari uji Saphiro-Wilk, kedua variables memiliki nilai pvalue kurang dari alpha (0.05) maka Tolak H0 dan data tidak berdistribusi normal. Oleh karena itu kita akan menggunakan Wilcoxon rank-sum test untuk menganalisa kedua data ini lebih dalam.

In [29]:

```
# Wilcoxon rank-sum test
from scipy.stats import ranksums
print('Two-tailed test : ', ranksums(Numerical_Variable['Customer Lifetime Value'], Numerical_Variable['Total Claim Amount']))
print('Right-tailed test : ', ranksums(Numerical_Variable['Customer Lifetime Value'], Numerical_Variable['Total Claim Amount'], alternative = 'greater'))
print('Left-tailed test : ', ranksums(Numerical_Variable['Customer Lifetime Value'], Numerical_Variable['Total Claim Amount'], alternative = 'less'))
```

```
Two-tailed test :  RanksumsResult(statistic=117.03468027466948, pvalue=0.0)
Right-tailed test :  RanksumsResult(statistic=117.03468027466948, pvalue=0.0)
Left-tailed test :  RanksumsResult(statistic=117.03468027466948, pvalue=1.0)
```

Key Points :

- **Pada Two-tailed test pvalue (0.0) kurang dari alpha (0.05) maka kita tolak H0 dikarenakan median dari kedua variable berbeda signifikan satu sama lain**
- **Pada Right-tailed test pvalue (0.0) kurang dari alpha (0.05) maka kita tolak H0 dikarenakan median dari variable pertama lebih besar daripada variable kedua**
- **Pada Left-tailed test pvalue (1.0) lebih dari alpha (0.05) maka kita tidak memiliki cukup bukti untuk menolak H0. Dikarenakan median populasi pertama tidak jauh lebih kecil dari median populasi kedua.**

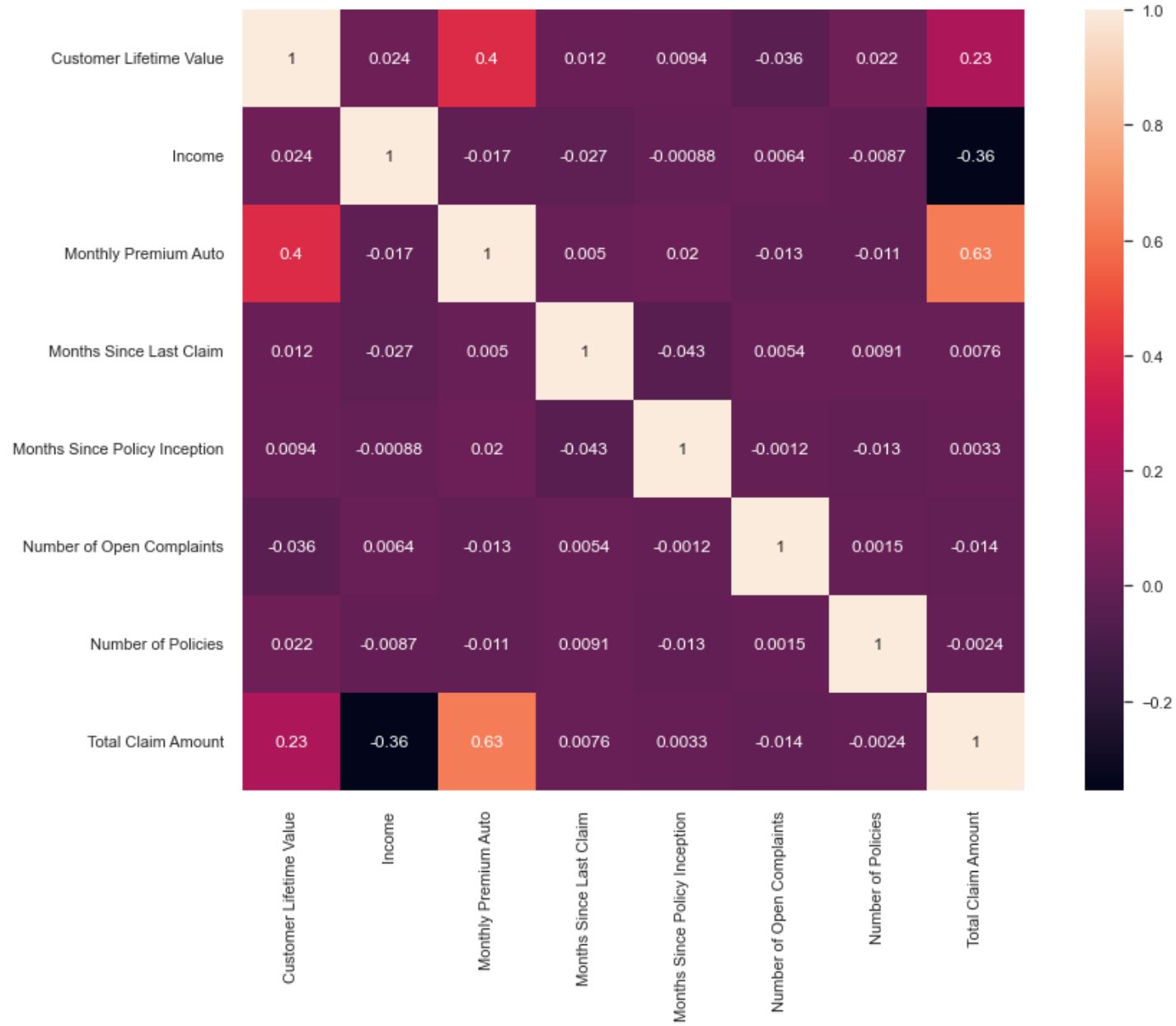
Dengan key point ketiga (Left-tailed test), kita mengetahui bahwa tidak ada data kita yang terlalu ekstrim (sudden spike). Oleh karena itu kita mengambil keputusan bahwa kita tidak akan membuang outliers yang ada. *Sumber 14*

Data Correlation

In [30]:

```
# Data Correlation with heatmap
```

```
# Data correlation with heatmap
sns.set(rc={'figure.figsize':(15,10)})
corr = df.corr()
sns.heatmap(corr,annot=True, square=True)
plt.show()
```



Strong correlations:

- **(Positive strong correlations) Monthly Premium Auto - Total Claim Amount**

Moderate correlations:

- **(Positive strong correlations) Monthly Premium Auto - Customer Lifetime Value**
- **(Negative strong correlations) Income - Total Claim Amount**

Sumber 15

Data Transformation

Pada tahap ini kita akan mentransformasi data agar dapat ditransformasikan ke dalam bentuk yang sesuai untuk data mining. Hal ini juga diperlukan untuk masuk ke dalam tahap Methodology (Modeling/Analysis).

In [31]:

```
# Change target variable 'Response' to 0 & 1
df['Response'] = pd.Series(map(lambda x: dict(Yes=1, No=0)[x],
                 df['Response'].values.tolist()), df['Response'].index)
```

In [32]:

```
## See all the unique data of dataset
listItem = []
for col in df.columns :
    listItem.append([col, df[col].dtype, df[col].isna().sum(), round((df[col].isna().sum()/len(df[col])) * 100,2),
                     df[col].nunique(), list(df[col].drop_duplicates().sample(2).values)]);

dfDesc = pd.DataFrame(columns=['dataFeatures', 'dataType', 'null', 'nullPct', 'unique', 'uniqueSample'],
                      data=listItem)
dfDesc
```

Out[32]:

	dataFeatures	dataType	null	nullPct	unique	uniqueSample
0	State	object	0	0.0	5	[Oregon, California]
1	Customer Lifetime Value	float64	0	0.0	8041	[4347.591949, 2827.328603]
2	Response	int64	0	0.0	2	[0, 1]
3	Coverage	object	0	0.0	3	[Premium, Basic]
4	Education	object	0	0.0	5	[Bachelor, Master]
5	EmploymentStatus	object	0	0.0	5	[Medical Leave, Employed]
6	Gender	object	0	0.0	2	[F, M]
7	Income	int64	0	0.0	5694	[96314, 67780]
8	Location Code	object	0	0.0	3	[Urban, Suburban]
9	Marital Status	object	0	0.0	3	[Married, Divorced]
10	Monthly Premium Auto	int64	0	0.0	202	[205, 67]
11	Months Since Last Claim	int64	0	0.0	36	[3, 32]
12	Months Since Policy Inception	int64	0	0.0	100	[52, 89]
13	Number of Open Complaints	int64	0	0.0	6	[5, 0]
14	Number of Policies	int64	0	0.0	9	[6, 9]
15	Policy Type	object	0	0.0	3	[Special Auto, Corporate Auto]
16	Policy	object	0	0.0	9	[Personal L2, Personal L3]
17	Renew Offer Type	object	0	0.0	4	[Offer2, Offer1]
18	Sales Channel	object	0	0.0	4	[Call Center, Branch]
19	Total Claim Amount	float64	0	0.0	5106	[396.295614, 139.841411]
20	Vehicle Class	object	0	0.0	6	[Sports Car, Four-Door Car]
21	Vehicle Size	object	0	0.0	3	[Medsize, Large]

Setelah merubah variable Response, hal selanjutnya adalah melakukan perubahan pada variable yang mempunyai numerical value yang cukup besar. Hal ini dilakukan karena nilai yang besar ini akan berpengaruh pada hasil perhitungan standardscaler (next step). Dimana pada standardscaler ini berkaitan dengan perhitungan nilai rata-rata. Dan jika dipikir secara matematis, nilai rata-rata pada variable yang mempunyai numerical value tinggi tentunya akan tetap lebih besar daripada nilai rata-rata pada variable yang mempunyai numerical value rendah dengan jumlah data yang sama. Oleh karena itu kita akan melakukan scaling natural-log pada variable untuk mengecilkan numerical value yang tinggi dan menghindari bias pada feature.

Dari data diatas, dapat kita lihat nilai mean antara variabel `Customer Lifetime Value`, `Income` dan `Total Claim Amount` terlampau berbanding jauh dengan variabel lainnya. Oleh karena itu kita akan melakukan scaling awal (Natural-Log) pada ketiga variable diatas.

Namun sebelum itu kita harus mengubah nilai 0 pada variable `Income` menjadi 1, dikarenakan log 0 =

undefined. Kenapa diisi dengan 1? dikarenakan log 1 = 0, maka hasil log akan tetap menjadi nilai awal. Hal ini juga berlaku pada variable `Total Claim Amount` dimana akan kita replace nilai > 1 dengan 1.

In [33]:

```
df1 = df.copy()
```

In [34]:

```
df1 = df1.astype({"Response": float})
```

In [35]:

```
# Replace Income Val 0 to 1
df1['Income'] = df1['Income'].replace(0,1)
df1.head()
```

Out[35]:

	State	Customer Lifetime Value	Response	Coverage	Education	EmploymentStatus	Gender	Income	Location Code	Marital Status	...	In
0	Washington	2763.519279	0.0	Basic	Bachelor	Employed	F	56274	Suburban	Married	...	
1	Arizona	6979.535903	0.0	Extended	Bachelor	Unemployed	F	1	Suburban	Single	...	
2	Nevada	12887.431650	0.0	Premium	Bachelor	Employed	F	48767	Suburban	Married	...	
3	California	7645.861827	0.0	Basic	Bachelor	Unemployed	M	1	Suburban	Married	...	
4	Washington	2813.692575	0.0	Basic	Bachelor	Employed	M	43836	Rural	Single	...	

5 rows × 22 columns

In [36]:

```
#create a new copy
df_clean = df1.copy()
```

In [37]:

```
# Transform to natural logarithm
df_clean['CLV_log'] = np.log(df_clean['Customer Lifetime Value'])
df_clean['Income_Log'] = np.log(df_clean['Income'])
df_clean['TCA_Log'] = np.log(df_clean['Total Claim Amount'])
```

In [38]:

```
# Remove original features
df_clean.drop(columns=['Customer Lifetime Value', 'Income', 'Total Claim Amount'], inplac
e=True)
```

In [39]:

```
# Previewed clean data
df_clean.head()
```

Out[39]:

	Monthly	Months	Numb

	State	Response	Coverage	Education	EmploymentStatus	Gender	Location Code	Marital Status	Monthly Premium Auto	Months Since Last Claim	...	Numb Polici
0	Washington	0.0	Basic	Bachelor	Employed	F	Suburban	Married	69	32	...	
1	Arizona	0.0	Extended	Bachelor	Unemployed	F	Suburban	Single	94	13	...	
2	Nevada	0.0	Premium	Bachelor	Employed	F	Suburban	Married	108	18	...	
3	California	0.0	Basic	Bachelor	Unemployed	M	Suburban	Married	106	18	...	
4	Washington	0.0	Basic	Bachelor	Employed	M	Rural	Single	73	12	...	

5 rows × 22 columns

In [40]:

```
# Previewed all columns
df_clean.columns
```

Out[40]:

```
Index(['State', 'Response', 'Coverage', 'Education', 'EmploymentStatus',
       'Gender', 'Location Code', 'Marital Status', 'Monthly Premium Auto',
       'Months Since Last Claim', 'Months Since Policy Inception',
       'Number of Open Complaints', 'Number of Policies', 'Policy Type',
       'Policy', 'Renew Offer Type', 'Sales Channel', 'Vehicle Class',
       'Vehicle Size', 'CLV_log', 'Income_Log', 'TCA_Log'],
      dtype='object')
```

Data Transform

In [41]:

```
# Preview clean data
df_clean.head()
```

Out[41]:

	State	Response	Coverage	Education	EmploymentStatus	Gender	Location Code	Marital Status	Monthly Premium Auto	Months Since Last Claim	...	Numb Polici
0	Washington	0.0	Basic	Bachelor	Employed	F	Suburban	Married	69	32	...	
1	Arizona	0.0	Extended	Bachelor	Unemployed	F	Suburban	Single	94	13	...	
2	Nevada	0.0	Premium	Bachelor	Employed	F	Suburban	Married	108	18	...	
3	California	0.0	Basic	Bachelor	Unemployed	M	Suburban	Married	106	18	...	
4	Washington	0.0	Basic	Bachelor	Employed	M	Rural	Single	73	12	...	

5 rows × 22 columns

Dari preview clean data diatas, kita dapat menentukan :

- **OneHotEncoder : Coverage, EmploymentStatus, Gender, Policy Type, Renew Offer Type, Sales Channel, Vehicle Size, State, Location Code,Marital Status**
- **Ordinal Encoder : Education**
- **Binary Encoder : Policy, Vehicle Class**
- **Robust Scaler : Customer Lifetime Value, Income, Monthly Premium Auto, Months Since Last Claim, Months Since Policy Inception, Number of Open Complaints, Number of Policies, Total Claim Amount**
  Untuk masing-masing transform tiap variable.

Namun disini kita akan membuat dua transformers yang berbeda. Dimana Logistic Regression dan KNN tidak akan menggunakan scaler dikarenakan model tersebut sensitive terhadap scaler dan untuk model Decision Tree and Random Forest tidak akan menggunakan scaler

In [42]:

```
# Import module for data transformation
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, RobustScaler, OrdinalEncoder
import category_encoders as ce
from category_encoders import BinaryEncoder
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
```

In [43]:

```
# Mapping variable education
ordinal_mapping = [
    {'col':'Education',
    'mapping':{
        'Doctor'  :    0,
        'Master'  :    1,
        'Bachelor':    2,
        'College' :    3,
        'High School or Below':    4
    }}
]
```

In [44]:

```
# Make transformer_scaled
transformer_scaled = ColumnTransformer([
    ('onehot', OneHotEncoder(handle_unknown='ignore'), ['Coverage','EmploymentStatus', 'G
ender', 'Policy Type', 'Renew Offer Type', 'Sales Channel', 'Vehicle Size', 'State', 'Lo
cation Code']),
    ('ordinal', ce.OrdinalEncoder(mapping=ordinal_mapping), ['Education']),
    ('binary', ce.BinaryEncoder(), ['Policy', 'Vehicle Class']),
    ('scaler', RobustScaler(), ['CLV_log', 'Income_Log','TCA_Log', 'Monthly Premium Auto'
, 'Months Since Last Claim', 'Months Since Policy Inception', 'Number of Open Complaints
', 'Number of Policies'])
])
```

In [45]:

```
# Make transformer_noscale
transformer_noscale = ColumnTransformer([
    ('onehot', OneHotEncoder(handle_unknown='ignore'), ['Coverage','EmploymentStatus', 'G
ender', 'Policy Type', 'Renew Offer Type', 'Sales Channel', 'Vehicle Size', 'State', 'Lo
cation Code']),
    ('ordinal', ce.OrdinalEncoder(mapping=ordinal_mapping), ['Education']),
    ('binary', ce.BinaryEncoder(), ['Policy', 'Vehicle Class'])
])
```

5. Methodology (Modeling/Analysis)

Data Splitting

In [46]:

```
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.metrics import accuracy_score, recall_score, plot_precision_recall_curve
```

In [47]:

```
X = df_clean.drop('Response', axis = 1)
y = df_clean['Response']
```

In [48]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify = y, test_size = 0.3,
random_state = 2022)
```

Benchmark Model without Handling Imbalance

Model yang digunakan dalam analysis data ini:

- **Logistic Regression**
- **KNeighbors**
- **Decision Tree**
- **Random Forest**

In [49]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [50]:

```
logreg = LogisticRegression(solver='lbfgs', max_iter=300, random_state = 2022)
tree = DecisionTreeClassifier(random_state = 2022)
knn = KNeighborsClassifier()
rfc = RandomForestClassifier()
```

In [51]:

```
logreg_pipe = Pipeline([
    ('transformer', transformer_scaled),
    ('model', logreg)
])

tree_pipe = Pipeline([
    ('transformer', transformer_noscale),
    ('model', tree)
])

knn_pipe = Pipeline([
    ('transformer', transformer_scaled),
    ('model', knn)
])

rfc_pipe = Pipeline([
    ('transformer', transformer_noscale),
    ('model', rfc)
])
```

In [52]:

```
from sklearn.metrics import accuracy_score, recall_score
```

In [53]:

```
def model_evaluation(model, metric):
    model_cv = cross_val_score(model, X_train, y_train, cv = StratifiedKFold(n_splits =
5), scoring = metric)
    return model_cv
```

Score dengan Accuracy

In [54]:

```
logreg_pipe_cv_ac = model_evaluation(logreg_pipe, 'accuracy')
tree_pipe_cv_ac = model_evaluation(tree_pipe, 'accuracy')
knn_pipe_cv_ac = model_evaluation(knn_pipe, 'accuracy')
rfc_pipe_cv_ac = model_evaluation(rfc_pipe, 'accuracy')
```

In [55]:

```
for model in [logreg_pipe, tree_pipe, knn_pipe, rfc_pipe]:
    model.fit(X_train, y_train)
```

In [56]:

```
score_cv = [logreg_pipe_cv_ac, tree_pipe_cv_ac, knn_pipe_cv_ac, rfc_pipe_cv_ac]
score_mean = [logreg_pipe_cv_ac.mean(), tree_pipe_cv_ac.mean(), knn_pipe_cv_ac.mean(), rf
c_pipe_cv_ac.mean()]
score_std = [logreg_pipe_cv_ac.std(), tree_pipe_cv_ac.std(), knn_pipe_cv_ac.std(), rfc_pi
pe_cv_ac.std()]
score_accuracy = [accuracy_score(y_test, logreg_pipe.predict(X_test)),
                  accuracy_score(y_test, tree_pipe.predict(X_test)),
                  accuracy_score(y_test, knn_pipe.predict(X_test)),
                  accuracy_score(y_test, rfc_pipe.predict(X_test))]
```

In [57]:

```
model_name = ['Logistic Regression', 'Decision Tree', 'KNN', 'RFC']
model_summary = pd.DataFrame({
    'model': model_name,
    'cv': score_cv,
    'mean cv': score_mean,
    'std cv': score_std,
    'accuracy_s': score_accuracy
})
model_summary
```

Out[57]:

	model	cv	mean cv	std cv	accuracy_s
0	Logistic Regression	[0.8686473807662236, 0.8717748240813136, 0.871...	0.870953	0.001389	0.872674
1	Decision Tree	[0.8631743549648163, 0.874120406567631, 0.8670...	0.870484	0.009562	0.875593
2	KNN	[0.8921032056293979, 0.893666927286943, 0.9007...	0.897702	0.009077	0.921926
3	RFC	[0.9030492572322126, 0.8983580922595777, 0.896...	0.897700	0.003353	0.902955

Score dengan Recall

In [58]:

```
logreg_pipe_cv_rc = model_evaluation(logreg_pipe, 'recall')
tree_pipe_cv_rc = model_evaluation(tree_pipe, 'recall')
knn_pipe_cv_rc = model_evaluation(knn_pipe, 'recall')
rfc_pipe_cv_rc = model_evaluation(rfc_pipe, 'recall')
```

In [59]:

```
score_cv_rc = [logreg_pipe_cv_rc, tree_pipe_cv_rc, knn_pipe_cv_rc, rfc_pipe_cv_rc]
score_mean_rc = [logreg_pipe_cv_rc.mean(), tree_pipe_cv_rc.mean(), knn_pipe_cv_rc.mean(),
```

```
rfc_pipe_cv_rc.mean()]
score_std_rc = [logreg_pipe_cv_rc.std(), tree_pipe_cv_rc.std(), knn_pipe_cv_rc.std(), rfc
_pipe_cv_rc.std()]
score_recall = [recall_score(y_test, logreg_pipe.predict(X_test)),
                recall_score(y_test, tree_pipe.predict(X_test)),
                recall_score(y_test, knn_pipe.predict(X_test)),
                recall_score(y_test, rfc_pipe.predict(X_test))]
```

In [60]:

```
model_name = ['Logistic Regression', 'Decision Tree', 'KNN', 'RFC']
model_summary_rc = pd.DataFrame({
    'model': model_name,
    'cv': score_cv_rc,
    'mean cv': score_mean_rc,
    'std cv': score_std_rc,
    'recall_s': score_recall
})
model_summary_rc
```

Out[60]:

	model	cv	mean cv	std cv	recall_s
0	Logistic Regression	[0.12021857923497267, 0.16393442622950818, 0.1...	0.140984	0.013996	0.180662
1	Decision Tree	[0.5846994535519126, 0.6120218579234973, 0.633...	0.627322	0.027518	0.651399
2	KNN	[0.6229508196721312, 0.6666666666666666, 0.677...	0.665574	0.021967	0.821883
3	RFC	[0.44808743169398907, 0.4371584699453552, 0.49...	0.445902	0.027344	0.501272

Handling Imbalance

Telah kita ketahui sebelumnya jika data kita imbalance sesuai dengan Response Yes dan Response No. Oleh karena itu kita harus membuat Handling Imbalance. Dimana hal ini dilakukan untuk membuat nilai dari `Accuracy` dan `Recall` kita lebih presisi dan tidak timpang sebelah.

In [61]:

```
# Count Response Yes & No
df.groupby(['Response'])['Response'].count()
```

Out[61]:

```
Response
0    7826
1    1308
Name: Response, dtype: int64
```

Terlihat bahwa berdasarkan target, data sangat timpang sebelah.

Secara umum, teknik sampling ada dua, yaitu Undersampling dan Oversampling. Adapun teknik yang diambil yaitu Oversampling. Dimana cara kerja dari teknik ini yaitu mengambil kelas minoritas sedemikian rupa (menduplikasi amatan minoritas) sehingga proporsinya dalam sample lebih besar dibandingkan proporsi asalnya. Diambilnya teknik ini dikarenakan untuk mencegah hilangnya data asli sample Response No dan setelah melakukan perbandingan, ternyata nilai `accuracy` dan `recall` pada oversampling lebih baik.

! Untuk menghemat code, kita akan memasukkan Random Over Sampling ini pada tahap modeling.

Benchmark Model with Random Over Sampling

In [62]:

```
logreg = LogisticRegression(solver='lbfgs', max_iter=300, random_state = 2022)
tree = DecisionTreeClassifier(random_state = 2022)
knn = KNeighborsClassifier()
```

```
rfc = RandomForestClassifier()
```

In [63]:

```
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler, NearMiss
```

In [64]:

```
ros = RandomOverSampler()
```

In [65]:

```
logreg_ros = LogisticRegression(solver='lbfgs', max_iter=300, random_state = 2022)
```

In [66]:

```
estimator_ros_logreg = Pipeline([
    ('transformer', transformer_scaled),
    ('balancing', ros),
    ('model', logreg_ros)
])

estimator_ros_knn = Pipeline([
    ('transformer', transformer_scaled),
    ('balancing', ros),
    ('model', knn)
])

estimator_ros_tree = Pipeline([
    ('transformer', transformer_noscale),
    ('balancing', ros),
    ('model', tree)
])

estimator_ros_rfc = Pipeline([
    ('transformer', transformer_noscale),
    ('balancing', ros),
    ('model', rfc)
])
```

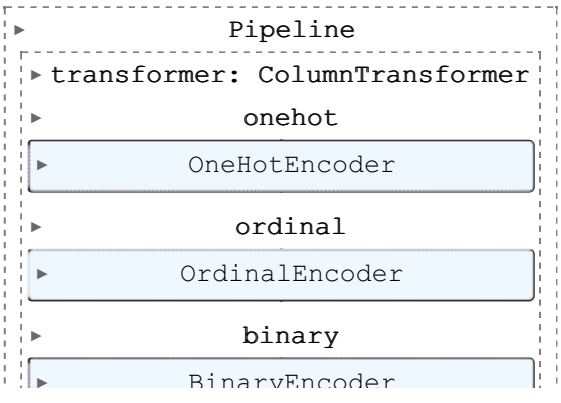
In [67]:

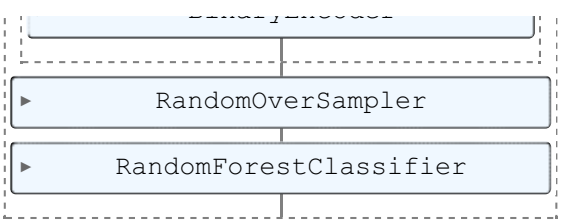
```
def model_evaluation_ros(model, metric):
    model_cv = cross_val_score(model, X_train, y_train, cv = StratifiedKFold(n_splits =
5), scoring = metric)
    return model_cv
```

In [68]:

```
# Pipeline
estimator_ros_logreg.fit(X_train, y_train)
estimator_ros_knn.fit(X_train, y_train)
estimator_ros_tree.fit(X_train, y_train)
estimator_ros_rfc.fit(X_train, y_train)
```

Out[68]:





Scoring by Accuracy

In [69]:

```
logreg_pipe_cv_ac_ros = model_evaluation_ros(estimator_ros_logreg, 'accuracy')
tree_pipe_cv_ac_ros = model_evaluation_ros(estimator_ros_tree, 'accuracy')
knn_pipe_cv_ac_ros = model_evaluation_ros(estimator_ros_knn, 'accuracy')
rfc_pipe_cv_ac_ros = model_evaluation_ros(estimator_ros_rfc, 'accuracy')
```

In [70]:

```
score_cv_ros_ac = [logreg_pipe_cv_ac_ros, tree_pipe_cv_ac_ros, knn_pipe_cv_ac_ros, rfc_pi
pe_cv_ac_ros]
score_mean_ros_ac = [logreg_pipe_cv_ac_ros.mean(), tree_pipe_cv_ac_ros.mean(), knn_pipe_c
v_ac_ros.mean(), rfc_pipe_cv_ac_ros.mean()]
score_std_ros_ac = [logreg_pipe_cv_ac_ros.std(), tree_pipe_cv_ac_ros.std(), knn_pipe_cv_a
c_ros.std(), rfc_pipe_cv_ac_ros.std()]
score_accuracy_ros_ac = [accuracy_score(y_test, estimator_ros_logreg.predict(X_test)),
                accuracy_score(y_test, estimator_ros_tree.predict(X_test)),
                accuracy_score(y_test, estimator_ros_knn.predict(X_test)),
                accuracy_score(y_test, estimator_ros_rfc.predict(X_test))]
```

In [71]:

```
model_name_ros_ac = ['Logistic Regression', 'Decision Tree', 'KNN', 'RFC']
model_summary_ros_ac = pd.DataFrame({
    'model': model_name_ros_ac,
    'cv': score_cv_ros_ac,
    'mean cv': score_mean_ros_ac,
    'std cv': score_std_ros_ac,
    'accuracy_ros': score_accuracy_ros_ac
})
model_summary_ros_ac
```

Out[71]:

	model	cv	mean cv	std cv	accuracy_ros
0	Logistic Regression	[0.671618451915559, 0.7075840500390931, 0.6825...	0.685906	0.012145	0.705582
1	Decision Tree	[0.8811571540265832, 0.8733385457388585, 0.871...	0.874863	0.011838	0.893834
2	KNN	[0.7709147771696638, 0.7795152462861611, 0.772...	0.775067	0.006237	0.787304
3	RFC	[0.8991399530883503, 0.9046129788897577, 0.903...	0.902550	0.005739	0.908063

Scoring by Recall

In [72]:

```
logreg_pipe_cv_rc_ros = model_evaluation_ros(estimator_ros_logreg, 'recall')
tree_pipe_cv_rc_ros = model_evaluation_ros(estimator_ros_tree, 'recall')
knn_pipe_cv_rc_ros = model_evaluation_ros(estimator_ros_knn, 'recall')
rfc_pipe_cv_rc_ros = model_evaluation_ros(estimator_ros_rfc, 'recall')
```

In [73]:

```
score_cv_ros_rc = [logreg_pipe_cv_rc_ros, tree_pipe_cv_rc_ros, knn_pipe_cv_rc_ros, rfc_pi
pe_cv_rc_ros]
score_mean_ros_rc = [logreg_pipe_cv_rc_ros.mean(), tree_pipe_cv_rc_ros.mean(), knn_pipe_c
v_rc_ros.mean(), rfc_pipe_cv_rc_ros.mean()]
score_std_ros_rc = [logreg_pipe_cv_rc_ros.std(), tree_pipe_cv_rc_ros.std(), knn_pipe_cv_r
```

```
c_ros.std(), rfc_pipe_cv_rc_ros.std()]
score_recall_ros_rc = [recall_score(y_test, estimator_ros_logreg.predict(X_test)),
                       recall_score(y_test, estimator_ros_tree.predict(X_test)),
                       recall_score(y_test, estimator_ros_knn.predict(X_test)),
                       recall_score(y_test, estimator_ros_rfc.predict(X_test))]
```

In [74]:

```
model_name_ros_rc = ['Logistic Regression', 'Decision Tree', 'KNN', 'RFC']
model_summary_ros_rc = pd.DataFrame({
    'model': model_name_ros_rc,
    'cv': score_cv_ros_rc,
    'mean cv': score_mean_ros_rc,
    'std cv': score_std_ros_rc,
    'recall_ros': score_recall_ros_rc
})
model_summary_ros_rc
```

Out[74]:

	model	cv	mean cv	std cv	recall_ros
0	Logistic Regression	[0.7158469945355191, 0.7158469945355191, 0.737...	0.732240	0.029325	0.770992
1	Decision Tree	[0.6612021857923497, 0.6666666666666666, 0.677...	0.657923	0.014081	0.689567
2	KNN	[0.9508196721311475, 0.9562841530054644, 0.989...	0.975956	0.018483	0.974555
3	RFC	[0.5628415300546448, 0.6284153005464481, 0.672...	0.607650	0.038547	0.689567

Dari data scoring yang sudah ada, terlihat bahwa model KNN adalah yang terbaik secara umum untuk dari setiap modelnya. Oleh karena itu kita akan menggunakan model KNN untuk di hyperparameter tuning

Hyperparameter Tuning

In [75]:

```
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
```

In [76]:

```
# Create a new pipeline for Hyperparameter Tuning
pipe = Pipeline([
        ('transformer', transformer_scaled),
        ('oversampling', ros),
        ('classifier', KNeighborsClassifier())
])
```

In [77]:

```
parameters = {
    'classifier__n_neighbors': range(1, 21, 2),
    'classifier__weights' : ['uniform','distance'],
    'classifier__metric' : ['minkowski','euclidean','manhattan']
}
```

In [78]:

```
gs = GridSearchCV(pipe, parameters, verbose = 1, cv=9, n_jobs = -1)
```

In [79]:

```
g_res = gs.fit(X_train, y_train)
```

```
Fitting 9 folds for each of 60 candidates, totalling 540 fits
```

In [80]:

```
g_res.best_score
```

```
g_res.best_score_
```

Out[80]:

```
0.900047762744619
```

In [81]:

```
g_res.best_params_
```

Out[81]:

```
{'classifier__metric': 'manhattan',
 'classifier__n_neighbors': 1,
 'classifier__weights': 'uniform'}
```

In [82]:

```
report_model = pd.DataFrame({'Accuracy': [0.785115], 'Recall': [0.974555], 'Tuned' : [0.
900047762744619]})
report_model
```

Out[82]:

	Accuracy	Recall	Tuned
0	**0.785115**	**0.974555**	**0.900048**

Dari nilai yang sudah didapat, ternyata hasil dari hyperparameter tuning dan model secara keseluruhan menunjukkan nilai Score KNN cukup baik. Dimana ketika dilakukan Hyperparameter Tuning, score dari model lebih baik performanya daripada score sebelumnya `Accuracy`. Oleh karena itu kita akan menggunakan model KNN yang sudah di tuned sebagai model akhir kita.

In [83]:

```
# Report Classification Report Tuned KNN
from sklearn.metrics import classification_report
y_pred_tuned = g_res.predict(X_test)

report_tuned = classification_report(y_test, y_pred_tuned)

print('Classification Report Tuned KNN : \n', report_tuned)
```

```
Classification Report Tuned KNN :
               precision    recall  f1-score   support

         0.0       0.99      0.89      0.94      2348
         1.0       0.60      0.96      0.73       393

    accuracy                           0.90      2741
   macro avg       0.79      0.92      0.84      2741
weighted avg       0.94      0.90      0.91      2741
```

Remodel data dengan best parameter

In [84]:

```
knn_tuned = KNeighborsClassifier(n_neighbors = 1, weights = 'uniform', metric = 'minkows
ki')
```

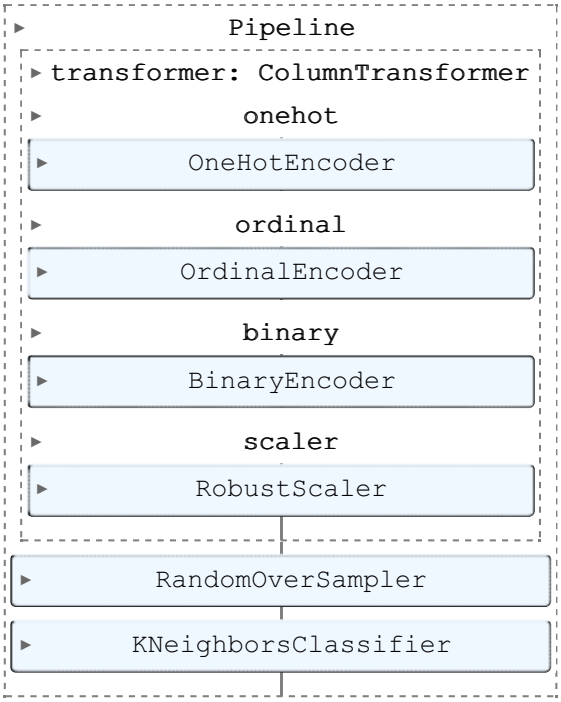
In [85]:

```
pipe_tuned = Pipeline([
        ('transformer', transformer_scaled),
        ('oversampling', ros),
        ('classifier', knn_tuned)
])
```

In [86]:

```
pipe_tuned.fit(X_train, y_train)
```

Out[86]:



In [87]:

```
y_hat = pipe_tuned.predict(X_train)
y_knn = pipe_tuned.predict(X_test)
```

In [88]:

```
print('Training set accuracy: ', metrics.accuracy_score(y_train, y_hat))
print('Test set accuracy: ',metrics.accuracy_score(y_test, y_knn))
```

```
Training set accuracy:  1.0
Test set accuracy:  0.903684786574243
```

In [89]:

```
print('Training set recall: ', metrics.recall_score(y_train, y_hat))
print('Test set recall: ',metrics.recall_score(y_test, y_knn))
```

```
Training set recall:  1.0
Test set recall:  0.9312977099236641
```

In [90]:

```
# Report Classification Report ReTuned with Best Parameter KNN
y_pred_retuned = g_res.predict(X_test)

report_retuned = classification_report(y_test, y_knn)

print('Classification Report ReTuned with Best Parameter KNN : \n', report_retuned)
```

```
Classification Report ReTuned with Best Parameter KNN :
               precision    recall  f1-score   support

         0.0       0.99      0.90      0.94      2348
         1.0       0.61      0.93      0.73       393

    accuracy                           0.90      2741
   macro avg       0.80      0.92      0.84      2741
weighted avg       0.93      0.90      0.91      2741
```

Terlihat bahwa dari report clasifikasi, model yang retuned memakai best parameter knn menjadi lebih seimbang di metric `Recall` nya. Oleh karena itu kita akan menggunakan model KNN yang sudah di retuned dengan best

parameter sebagai model akhir kita.

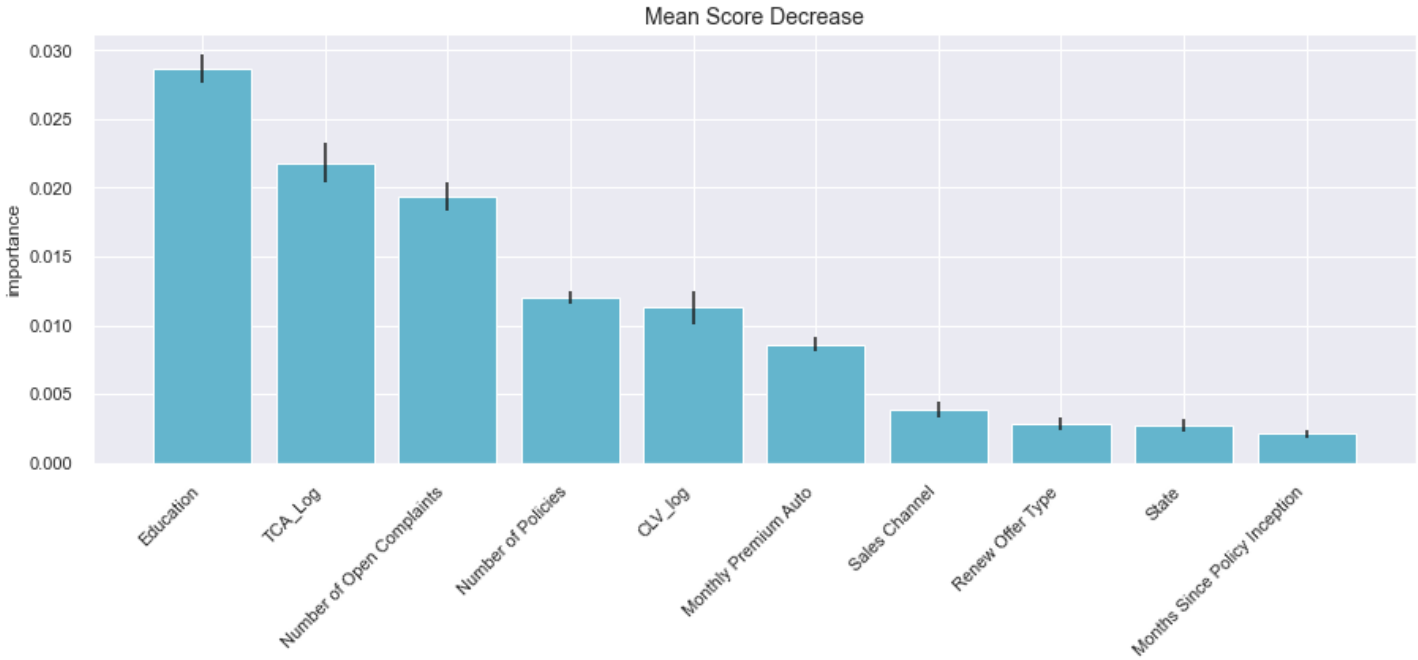
Feature Importance

In [91]:

```
from jcopml.feature_importance import mean_score_decrease
```

In [92]:

```
# Feature Importance
df_imp = mean_score_decrease(X_train, y_train, pipe_tuned, plot=True, topk=10)
```



Terlihat bahwa ternyata untuk model KNN kita, fitur/kolom Education adalah yang paling penting. Kemudian diikuti dengan Total Claim Amount, Number of Open Complaints, Number of Policies, CLV_log dan selanjutnya.

6. Conclusion and Recommendation

Conclusion

In [93]:

```
print('Classification Report ReTuned with Best Parameter KNN : \n', report_retuned)
```

```
Classification Report ReTuned with Best Parameter KNN :
              precision    recall  f1-score   support

         0.0       0.99      0.90      0.94      2348
         1.0       0.61      0.93      0.73       393

    accuracy                           0.90      2741
   macro avg       0.80      0.92      0.84      2741
weighted avg       0.93      0.90      0.91      2741
```

Berdasarkan hasil classification report dari model kita, kita dapat menyimpulkan bahwa jika seandainya nanti kita menggunakan model kita untuk memfilter calon customer yang akan mendapatkan response yes, maka model kita dapat memprediksi response yes sebanyak 93% dari keseluruhan calon customer yang sebenarnya mendapatkan response yes dan mengurangi 90% calon customer yang mendapatkan response no untuk kita tidak approach (recall).

Model kita ini memiliki ketepatan untuk memprediksi response yes sebesar 61% (precisionnya). Jadi apabila kita memprediksi bahwa calon customer akan mendapatkan response yes, maka kemungkinan tebakan benarnya sebesar 61% kurang lebih. Tetapi sebenarnya masih akan ada response calon customer yang sebenarnya mendapatkan response no namun diprediksi sebagai calon customer yang mendapatkan response yes sekitar 10% dari keseluruhan calon customer yang mendapatkan response no (berdasarkan recall).

Bila seandainya biaya untuk approach per calon customer itu 1$. Dan andaikan jumlah calon customer yang kita miliki untuk suatu kurun waktu sebanyak 200 orang (dimana andaikan 100 orang memiliki response yes, dan 100 orang lagi mendapatkan response no), maka perbandingannya kurang lebih akan seperti ini :

Tanpa Model (semua kandidat kita check dan tawarkan) :

Total Biaya => 200 x 1 USD = 200 USD
Total calon customer dengan response yes => 100 orang (karena semua kita tawarkan)
Total calon customer dengan response yes namun tidak didapatkan => 0 orang (karena semua kita tawarkan)
Total calon customer dengan response no => 100
Biaya yang terbuang => 100 x 1 USD = 100 USD (karena 100 orang menolak dan menjadi sia-sia)
Jumlah penghematan => 0 USD

Dengan Model (hanya kandidat yang diprediksi oleh model tertarik yang kita check dan tawarkan) :

Total Biaya => (93 x 1 USD) + (10 x 1 USD) = 93 USD + 10 USD = 103 USD
Total calon customer dengan response yes => 93 orang (karena recall 1 response yes 93%)
Total calon customer dengan response yes namun di predisksi no => 7 orang (karena recall 1 response no 93%)
Biaya yang terbuang => 10 x 1 USD = 10 USD (berdasarkan recall 0 response yes)
Jumlah penghematan => 90 x 1 USD = 90 USD (yang dihitung pure hanya yang response no)

Berdasarkan contoh hitungan tersebut, terlihat bahwa dengan menggunakan model kita, maka perusahaan tersebut akan menghemat biaya yang cukup besar 90/200 atau sebanyak 45% pengeluaran marketing tanpa mengorbankan terlalu banyak jumlah calon customer yang mendapatkan response yes (yang dihitung pure hanya yang response no).

Recommendation

Hal-hal yang bisa dilakukan untuk mengembangkan project dan modelnya lebih baik lagi :

- `Business` **:**
  - **Dari sisi bisnis, tim marketing pada perusahaan bisa untuk membuat promo berdua atau bundling (paket) yang lebih oke dan ciamik, misalkan dengan membuat promo pembelian kedua discount 50%. Dimana hal ini sesuai dengan market share yang telah kita lihat sebelumnya bahwa banyak calon customer yang berstatus sudah menikah. Dengan agregasi response yes yang lebih banyak, tentunya perusahaan lebih aman untuk marketing pada lini bisnis ini.**
  - **Selain itu perusahaan juga dapat melakukan marketing yang lebih gencar kepada calon customer yang bertempat tinggal di wilayah suburban. Karena dari segi jumlah dan agregasi response yes, wilayah suburban sangat mendominasi di setiap wilayah bagiannya. Atau juga boleh untuk lebih melakukan marketing di wilayah urban dan rural dengan menganalisa terlebih dahulu pola calon customer yang mendapatkan response yes di wilayah ini.**
  - **Melihat dari persebaran penjualan polis dan agregasi response yes, perusahaan bisa untuk memfokuskan marketingnya pada polis tipe personal. Dimana perusahaan dapat membuat berbagai macam tipe pada polis personal ini**
  - **Untuk menambah penjualan polis tipe company, perusahaan dapat melakukan kolaborasi dengan perusahaan lain seperti bank atau perusahaan lainnya. Dengan tujuan agar karyawan pada perusahaan tersebut melakukan pembayaran premi pada perusahaan ini.**
  - **Perusahaan dapat mereview kembali Renew Offer Type `Offer3` dan `Offer4` karena pada persentase calon customer yang di response yes sangat sedikit. `Offer3` hanya memiliki persentase yes sebanyak 2% sedangkan `Offer4` sama sekali tidak memiliki response yes.**

- `Project Model` **:**
  - **Mencoba algorithm ML yang lain seperti membuat clustering dari behaviour calon customer, sehingga perusahaan mudah untuk mengkategorisasikan calon customer mereka sesuai dengan variable yang ada**

- **Mencoba hyperparameter tuning kembali, menggunakan teknik oversampling yang berbeda juga selain Random Over Sampling, seperti SMOTENC, dll**
- **Menganalisa data model lebih lanjut yang kemungkinannya kita masih salah tebak untuk mengetahui alasannya dan karakteristiknya sehingga dapat menyempurnakan project dan model kita kedepannya**
- **Menambahkan fitur-fitur (variable-variable) baru yang kemungkinan ada korelasinya dengan dataset kita miliki. Dimana seperti yang kita tahu bahwa banyak sekali preliminary Car Insurance Company dalam mengambil keputusan untuk menerima calon nasabahnya seperti umur dari calon customer, umur kendaraan dari calon customer, driving record dari calon customer dll. Hal ini dimaksudkan untuk memastikan apakah calon customer riskan dan dirasa akan merugikan company atau tidak**