premium_estimator_main

June 16, 2025

[45]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split,KFold,␣
 ↪cross_val_score,cross_validate,RandomizedSearchCV
from sklearn.metrics import mean_squared_error,r2_score,root_mean_squared_error
from sklearn.linear_model import LinearRegression,Lasso,Ridge
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from time import time
```

0.1 Data Import & Exploration

[46]:
```
# Switching to the source directory

parent_dir = os.path.abspath(os.path.join(os.getcwd(), "../"))
parent_dir
print(parent_dir)
```

```
C:\Users\91948\Downloads\BKs\Projects\personalized_health_insurance_premium_esti
mator
```

[47]:
```
# Reading the data

df = pd.read_excel(fr'{parent_dir}\data\premiums.xlsx',sheet_name='Sheet1')
```

[48]:
```
# Displaying the data

df.head()
```

[48]:

	Age	Gender	Region	Marital_status	Number Of Dependants	BMI_Category	\
0	26	Male	Northwest	Unmarried	0	Normal	
1	29	Female	Southeast	Married	2	Obesity	
2	49	Female	Northeast	Married	2	Normal	

```
3   30  Female  Southeast        Married                     3       Normal
4   18    Male  Northeast      Unmarried                     0   Overweight

  Smoking_Status Employment_Status Income_Level  Income_Lakhs  \
0     No Smoking          Salaried         <10L             6
1        Regular          Salaried         <10L             6
2     No Smoking     Self-Employed    10L - 25L            20
3     No Smoking          Salaried        > 40L            77
4        Regular     Self-Employed        > 40L            99

       Medical History Insurance_Plan  Annual_Premium_Amount
0             Diabetes         Bronze                   9053
1             Diabetes         Bronze                  16339
2  High blood pressure         Silver                  18164
3           No Disease           Gold                  20303
4  High blood pressure         Silver                  13365
```

[49]:
```
df.shape
```

[49]:
```
(50000, 13)
```

[50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Age                    50000 non-null  int64
 1   Gender                 50000 non-null  object
 2   Region                 50000 non-null  object
 3   Marital_status         50000 non-null  object
 4   Number Of Dependants   50000 non-null  int64
 5   BMI_Category           50000 non-null  object
 6   Smoking_Status         49989 non-null  object
 7   Employment_Status      49998 non-null  object
 8   Income_Level           49987 non-null  object
 9   Income_Lakhs           50000 non-null  int64
 10  Medical History        50000 non-null  object
 11  Insurance_Plan         50000 non-null  object
 12  Annual_Premium_Amount  50000 non-null  int64
dtypes: int64(4), object(9)
memory usage: 5.0+ MB
```

0.2 Data Cleaning

0.2.1 Column Formatter

[51]:
```
# Renaming the columns with proper formatter
# Eg: Number Of Dependants -> number_of_dependants

df.columns = df.columns.str.replace(' ','_').str.lower()
```

[52]:
```
df.head()
```

[52]:

	age	gender	region	marital_status	number_of_dependants	bmi_category	\
0	26	Male	Northwest	Unmarried	0	Normal	
1	29	Female	Southeast	Married	2	Obesity	
2	49	Female	Northeast	Married	2	Normal	
3	30	Female	Southeast	Married	3	Normal	
4	18	Male	Northeast	Unmarried	0	Overweight	

	smoking_status	employment_status	income_level	income_lakhs	\
0	No Smoking	Salaried	<10L	6	
1	Regular	Salaried	<10L	6	
2	No Smoking	Self-Employed	10L - 25L	20	
3	No Smoking	Salaried	> 40L	77	
4	Regular	Self-Employed	> 40L	99	

	medical_history	insurance_plan	annual_premium_amount
0	Diabetes	Bronze	9053
1	Diabetes	Bronze	16339
2	High blood pressure	Silver	18164
3	No Disease	Gold	20303
4	High blood pressure	Silver	13365

0.2.2 Handling Missing Values

[53]:
```
# Finding the columns with NA values

df.isna().sum()
```

[53]:
```
age                     0
gender                  0
region                  0
marital_status          0
number_of_dependants    0
bmi_category            0
smoking_status         11
employment_status       2
income_level           13
income_lakhs            0
```

```
medical_history           0
insurance_plan            0
annual_premium_amount     0
dtype: int64
```

[54]:
```
# Dropping the Rows where it has NA value

# Before Dropping
df[df['smoking_status'].isna()].head()
```

[54]:
```
       age  gender     region marital_status  number_of_dependants  \
177     26  Female  Southwest        Married                     3
15648   47    Male  Southwest        Married                     4
16324   45    Male  Northwest        Married                     4
16941   34    Male  Southwest        Married                     5
16975   23    Male  Southwest      Unmarried                     0

      bmi_category smoking_status employment_status income_level  \
177    Underweight            NaN          Salaried        > 40L
15648       Normal            NaN        Freelancer    25L - 40L
16324   Overweight            NaN          Salaried    10L - 25L
16941       Normal            NaN     Self-Employed    25L - 40L
16975       Normal            NaN        Freelancer         <10L

       income_lakhs                      medical_history insurance_plan  \
177              69                             Diabetes           Gold
15648            32                             Diabetes           Gold
16324            16  High blood pressure & Heart disease         Silver
16941            35  High blood pressure & Heart disease           Gold
16975             3                           No Disease         Bronze

       annual_premium_amount
177                    22605
15648                  26100
16324                  21881
16941                  25865
16975                   6001
```

[55]:
```
# Dropping
df.dropna(how='any',inplace = True)
```

[56]:
```
# After Dropping
df[df['smoking_status'].isna()]
```

[56]:
```
Empty DataFrame
Columns: [age, gender, region, marital_status, number_of_dependants,
bmi_category, smoking_status, employment_status, income_level, income_lakhs,
```

```
medical_history, insurance_plan, annual_premium_amount]
Index: []
```

[57]:
```
df.isna().sum()
```

[57]:
```
age                     0
gender                  0
region                  0
marital_status          0
number_of_dependants    0
bmi_category            0
smoking_status          0
employment_status       0
income_level            0
income_lakhs            0
medical_history         0
insurance_plan          0
annual_premium_amount   0
dtype: int64
```

[58]:
```
df.reset_index(inplace=True,drop=True)
```

0.2.3 Handling Duplicated Rows

[59]:
```
# Duplicated rows

df[df.duplicated()]
```

[59]:
```
Empty DataFrame
Columns: [age, gender, region, marital_status, number_of_dependants,
bmi_category, smoking_status, employment_status, income_level, income_lakhs,
medical_history, insurance_plan, annual_premium_amount]
Index: []
```

Here we dont have any duplicated rows but for the safe purpose we are dropping them

[60]:
```
# Dropping the duplicated rows

df.drop_duplicates(inplace=True)
```

0.2.4 Fomatting Values - Numeric Columns

[61]:
```
df.dtypes
```

[61]:
```
age                      int64
gender                  object
region                  object
marital_status          object
```

```
number_of_dependants      int64
bmi_category             object
smoking_status           object
employment_status        object
income_level             object
income_lakhs              int64
medical_history          object
insurance_plan           object
annual_premium_amount     int64
dtype: object
```

[62]:
```
# Selecting only the columns with numerical values

numeric_columns = df.select_dtypes(include=['int64']).columns
numeric_columns
```

[62]:
```
Index(['age', 'number_of_dependants', 'income_lakhs', 'annual_premium_amount'],
dtype='object')
```

[63]:
```
# Printing the unique values of each numeric columns to identify the values␣
 ↪with improper formats

for col in numeric_columns:
    print(f'{col}:\n',df[col].unique())
    print('*'*100)
```

```
age:
 [ 26  29  49  30  18  56  33  43  59  22  21  46  68  60  27  25  36  20
  28  32  19  55  35  52  40  23  50  41  67  37  24  34  54  42  45  44
  57  38  31  58  48  51 224  47  39  53  66  64  65  62  61  70  72  69
  71 124  63 136 203 356]
****************************************************************************
********************
number_of_dependants:
 [ 0  2  3  4  1  5 -3 -1]
****************************************************************************
********************
income_lakhs:
 [  6  20  77  99  14   4  46  21   3  97   1  27  15  18   7  37  30  13
   8  83  19  29   5  70  11  33  23  40  84  22   9  71  59  38  35  28
  39  57  25  12  36  92   2  24  16  34  93  78  26  49  68  52  62  31
  90  50  32  10  88  54  86  41  95  64  85  81  79  56  80  17  98  89
  82 100  44  66  53  75  94  69  58  74  65  91  42  61  87  96  51  67
  43  73  63  55  48  45  47  72  60 560  76 440 630 900 930 580 700 790
 770 680]
****************************************************************************
********************
annual_premium_amount:
```

```
 [ 9053 16339 18164 … 26370 10957 27076]
********************************************************************************
********************
```

We have negative values in `number_of_dependents`. This is not appropriate. Hence we will handle these values by making the negative values as absolute values

[64]:
```
# Before formatting

df['number_of_dependants'].unique()
```

[64]:
```
array([ 0,  2,  3,  4,  1,  5, -3, -1])
```

[65]:
```
df['number_of_dependants'] = abs(df['number_of_dependants'])
```

[66]:
```
# After formatting

df['number_of_dependants'].unique()
```

[66]:
```
array([0, 2, 3, 4, 1, 5])
```

0.2.5 Fomatting Values - Categorical Columns

[67]:
```
# Selecting only the columns with categorical values

cat_columns = df.select_dtypes(include=['object']).columns
cat_columns
```

[67]:
```
Index(['gender', 'region', 'marital_status', 'bmi_category', 'smoking_status',
       'employment_status', 'income_level', 'medical_history',
       'insurance_plan'],
      dtype='object')
```

[68]:
```
# Printing the unique values of each categorical columns to identify the values␣
↪with improper formats

for col in cat_columns:
    print(f'{col}:\n',list(df[col].unique()))
    print('*'*100)
```

```
gender:
 ['Male', 'Female']
********************************************************************************
********************
region:
 ['Northwest', 'Southeast', 'Northeast', 'Southwest']
********************************************************************************
********************
marital_status:
```

```
 ['Unmarried', 'Married']
**********************************************************************
********************
bmi_category:
 ['Normal', 'Obesity', 'Overweight', 'Underweight']
**********************************************************************
********************
smoking_status:
 ['No Smoking', 'Regular', 'Occasional', 'Smoking=0', 'Does Not Smoke', 'Not
Smoking']
**********************************************************************
********************
employment_status:
 ['Salaried', 'Self-Employed', 'Freelancer']
**********************************************************************
********************
income_level:
 ['<10L', '10L - 25L', '> 40L', '25L - 40L']
**********************************************************************
********************
medical_history:
 ['Diabetes', 'High blood pressure', 'No Disease', 'Diabetes & High blood
pressure', 'Thyroid', 'Heart disease', 'High blood pressure & Heart disease',
'Diabetes & Thyroid', 'Diabetes & Heart disease']
**********************************************************************
********************
insurance_plan:
 ['Bronze', 'Silver', 'Gold']
**********************************************************************
********************
```

As per the above result, we can see that `smoking_status` column has multiple values with same meaning. Hence we will format this column

[69]:
```
# Before formatting

df['smoking_status'].unique()
```

[69]:
```
array(['No Smoking', 'Regular', 'Occasional', 'Smoking=0',
       'Does Not Smoke', 'Not Smoking'], dtype=object)
```

[70]:
```
# Replacing values with desired formats

df['smoking_status'] = df['smoking_status'].replace(
    {
        'Smoking=0' : 'No Smoking',
         'Does Not Smoke' : 'No Smoking',
         'Not Smoking' : 'No Smoking',
```

```
    }
)
```

[71]:
```
# After formatting

df['smoking_status'].unique()
```

[71]:
```
array(['No Smoking', 'Regular', 'Occasional'], dtype=object)
```

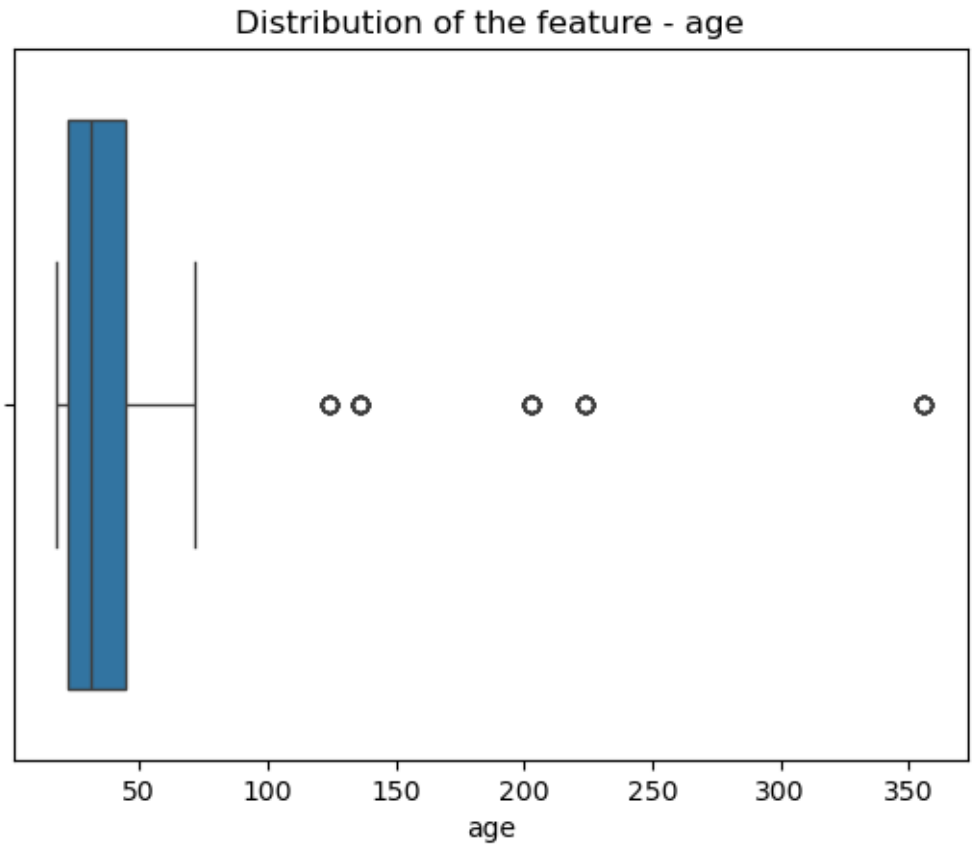
0.2.6 Outlier Treatment

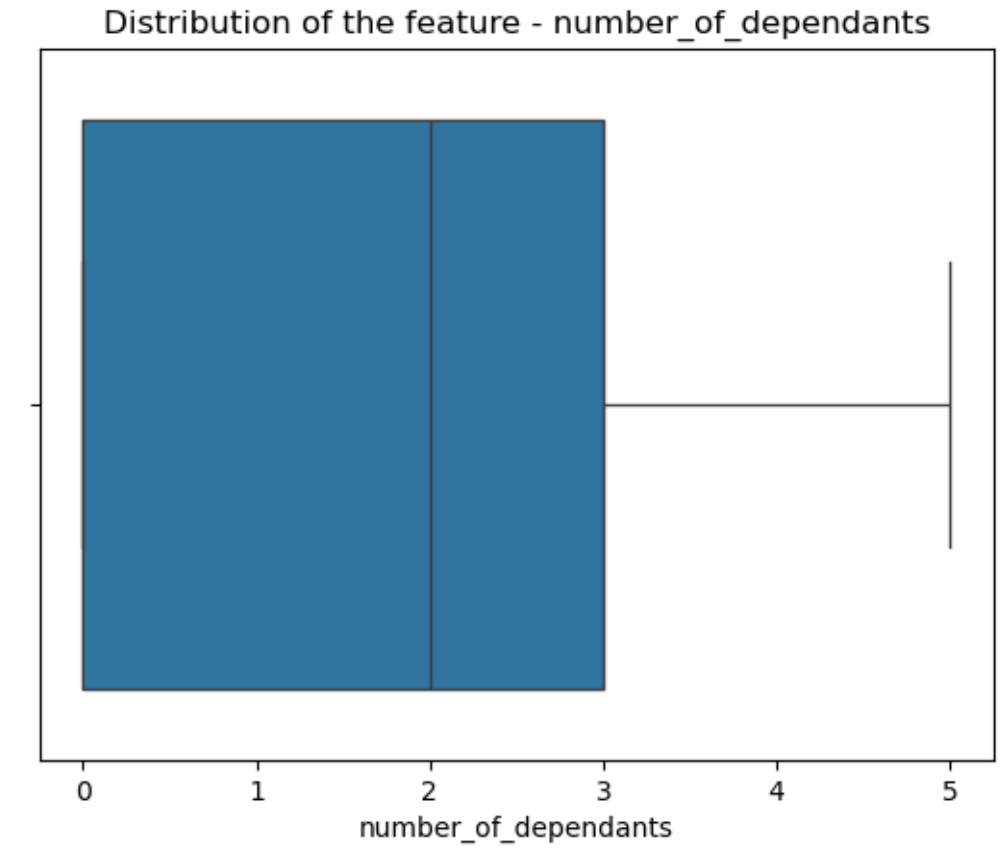
[72]:
```
df.describe()
```

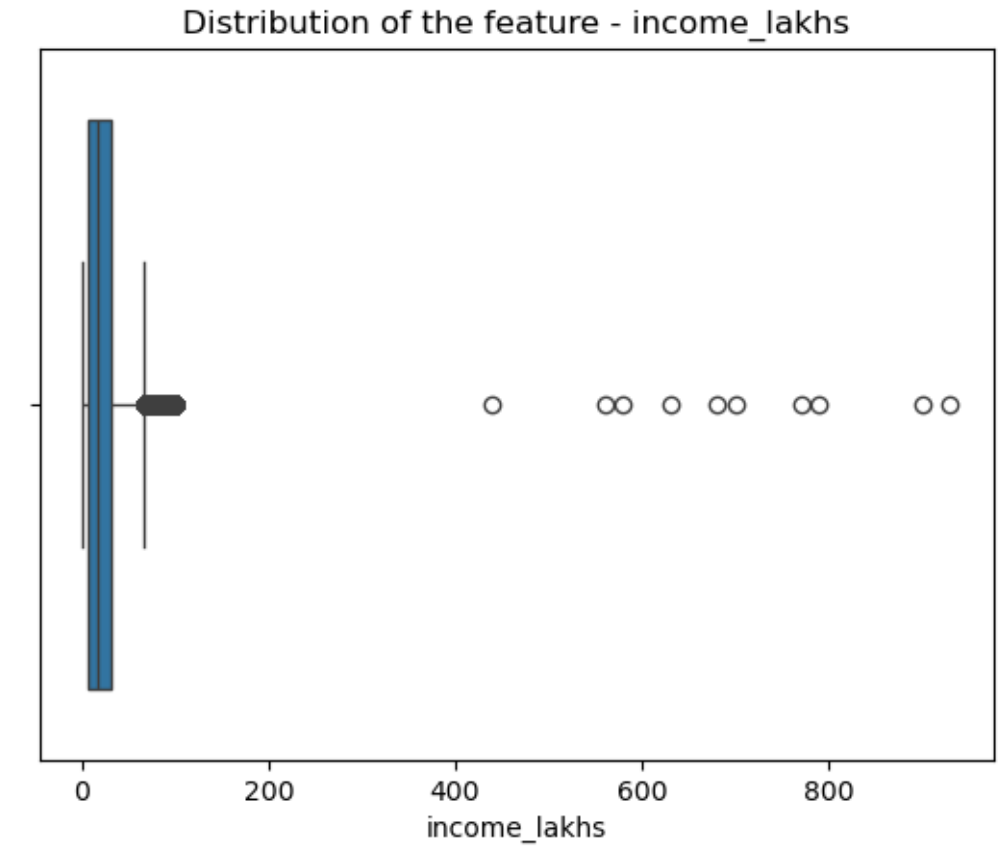
[72]:

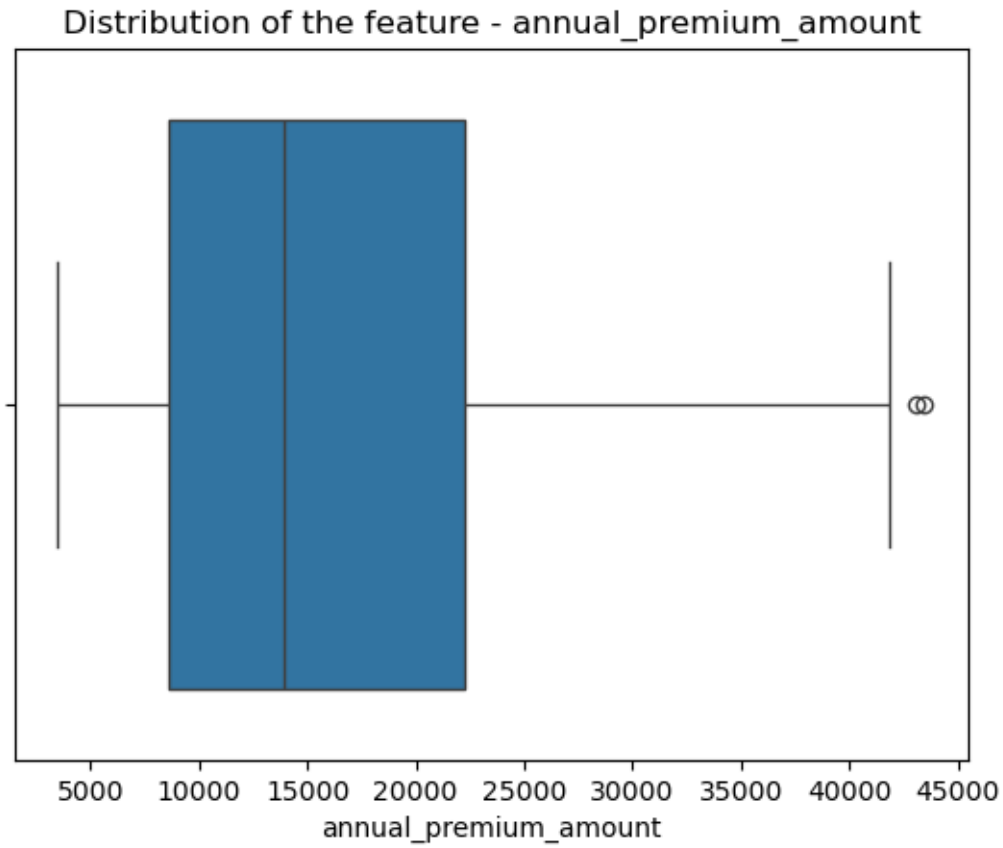
	age	number_of_dependants	income_lakhs	annual_premium_amount
count	49976.000000	49976.000000	49976.000000	49976.000000
mean	34.591764	1.717284	23.021150	15766.810189
std	15.000378	1.491953	24.221794	8419.995271
min	18.000000	0.000000	1.000000	3501.000000
25%	22.000000	0.000000	7.000000	8607.750000
50%	31.000000	2.000000	17.000000	13928.000000
75%	45.000000	3.000000	31.000000	22273.500000
max	356.000000	5.000000	930.000000	43471.000000

[73]:
```
# Plotting box plot for each numerical columns to detect outliers

for col in numeric_columns:
    sns.boxplot(data=df,x=col)
    plt.title(f'Distribution of the feature - {col}')
    plt.show()
```


Distribution of the feature - age
50
100
150
200
250
300
350
age


Distribution of the feature - number_of_dependants
0
1
2
3
4
5
number_of_dependants


Distribution of the feature - income_lakhs
0
200
400
600
800
income_lakhs



As we can see above, `age` & `income_lakhs` columns have outliers. We will see it using the above box plot also

Age

[74]:
```
# Before
df['age'].unique()
```

[74]:
```
array([ 26,  29,  49,  30,  18,  56,  33,  43,  59,  22,  21,  46,  68,
        60,  27,  25,  36,  20,  28,  32,  19,  55,  35,  52,  40,  23,
        50,  41,  67,  37,  24,  34,  54,  42,  45,  44,  57,  38,  31,
        58,  48,  51, 224,  47,  39,  53,  66,  64,  65,  62,  61,  70,
        72,  69,  71, 124,  63, 136, 203, 356])
```

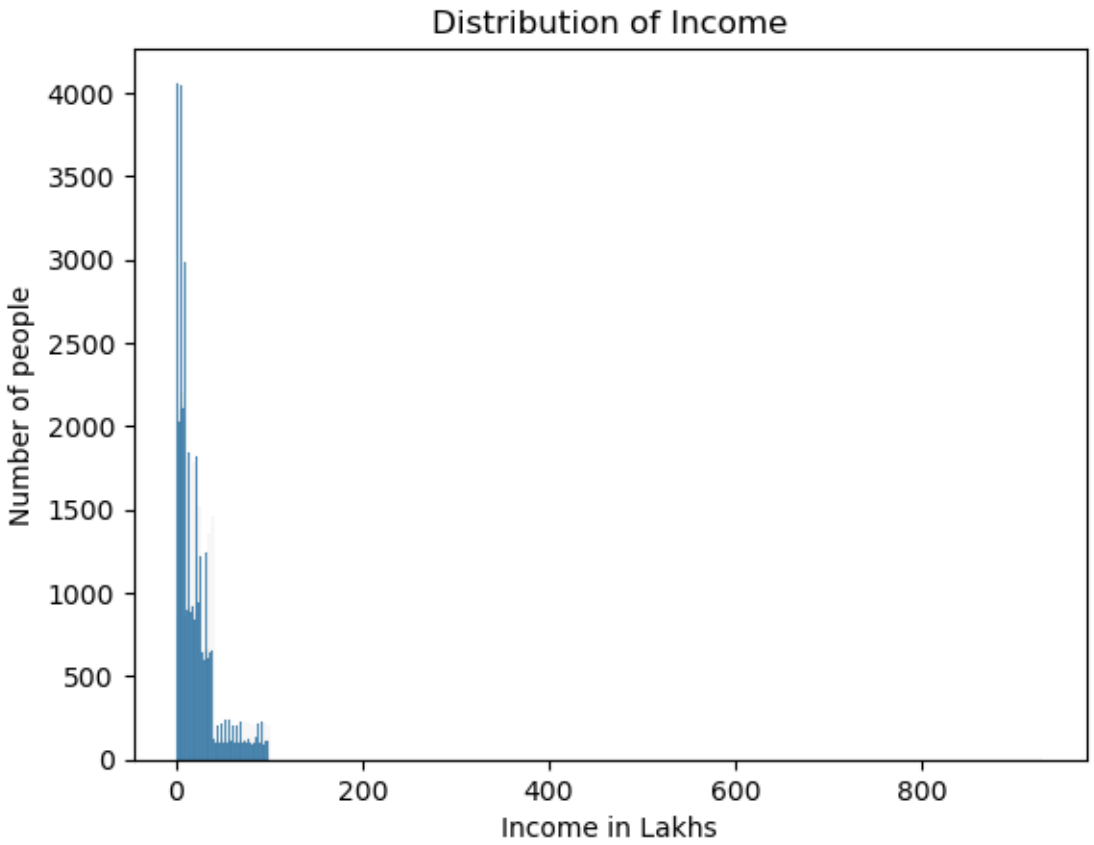
[75]:
```
# Selecting only the rows where age <= 100. Because age > 100 is outlier

df1 = df[df['age'] <= 100]
df1.reset_index(inplace=True,drop=True)
```

[76]:
```
# After
df1['age'].unique()
```

[76]:
```
array([26, 29, 49, 30, 18, 56, 33, 43, 59, 22, 21, 46, 68, 60, 27, 25, 36,
       20, 28, 32, 19, 55, 35, 52, 40, 23, 50, 41, 67, 37, 24, 34, 54, 42,
       45, 44, 57, 38, 31, 58, 48, 51, 47, 39, 53, 66, 64, 65, 62, 61, 70,
       72, 69, 71, 63])
```

Income

[77]:
```
# Distribution of Income using Histogram

sns.histplot(data=df1,x='income_lakhs')
plt.title('Distribution of Income')
plt.xlabel('Income in Lakhs')
plt.ylabel('Number of people')
plt.show()
```



[78]:
```
# Determining Lower Salary Boundary and Upper Salary Boundary using IQR Method

# Quartile 1 and Quartile 3

q1 = np.percentile(df1['income_lakhs'],25)
q3 = np.percentile(df1['income_lakhs'],75)
```

```
print(f'Q1 -> {q1}\nQ3 -> {q3}')

# Inter Qurartile Range

iqr = q3-q1
print('IQR ->',iqr)

# Lower and Upper boundary using IQR

lower_boundary = q1 - (iqr * 1.5)
upper_boundary = q3 + (iqr * 1.5)
print(f'Lower Boundary -> {lower_boundary}\nUpper Boundary -> {upper_boundary}')
```

```
Q1 -> 7.0
Q3 -> 31.0
IQR -> 24.0
Lower Boundary -> -29.0
Upper Boundary -> 67.0
```

NOTE:

Our upper boundary is too low. Hence we will check with business and determine the optimal upper boundary to find the outliers in income_lakhs column

We decided that the upper boundary will be 1crore i.e 100 lakhs. Whatever more than that are outliers

[79]:
```
# Selecting only the rows where income <= 100 lakhs. Because income > 100 lakhs␣
↪is outlier

income_threshold = 100
df2 = df1[df1['income_lakhs'] <= income_threshold]
```

[80]:
```
df2.reset_index(drop=True,inplace=True)
```

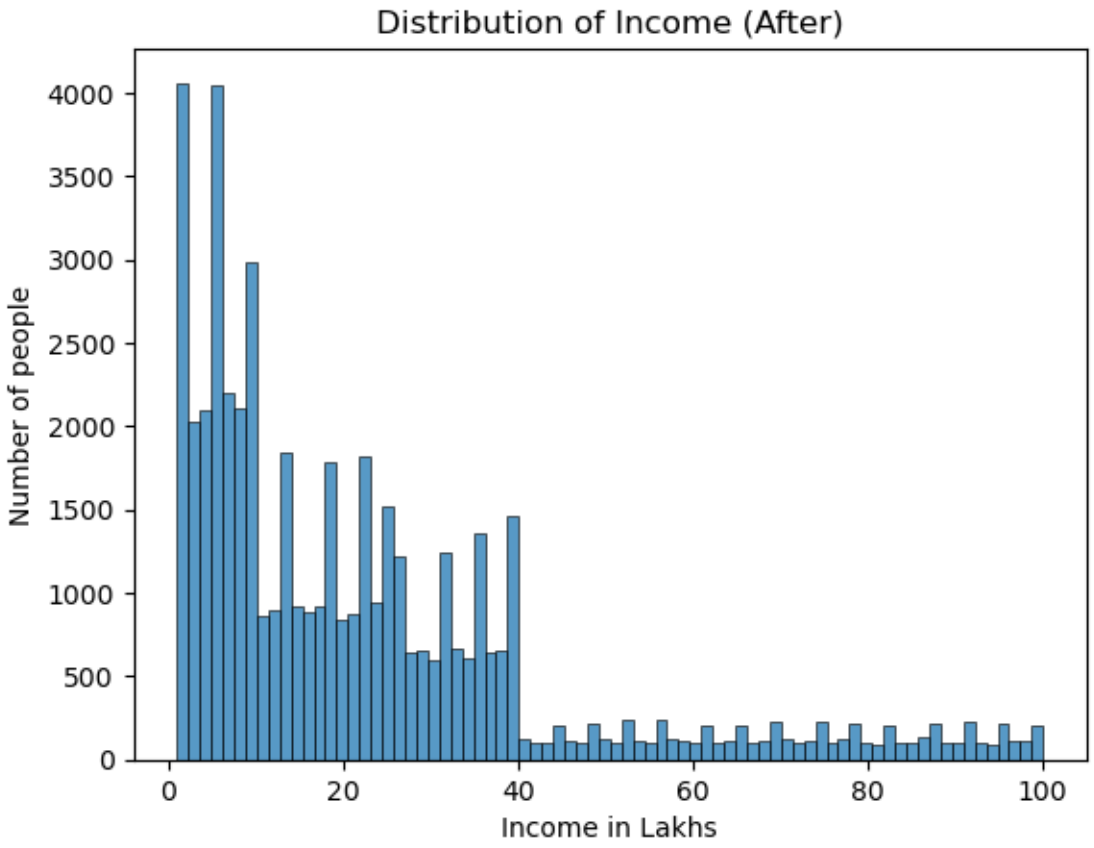
[81]:
```
df2
```

[81]:

	age	gender	region	marital_status	number_of_dependants	\
0	26	Male	Northwest	Unmarried	0	
1	29	Female	Southeast	Married	2	
2	49	Female	Northeast	Married	2	
3	30	Female	Southeast	Married	3	
4	18	Male	Northeast	Unmarried	0	
…	…	…	…	…	…	
49903	24	Female	Northwest	Unmarried	0	
49904	47	Female	Southeast	Married	2	
49905	21	Male	Northwest	Unmarried	0	
49906	18	Male	Northwest	Unmarried	2	
49907	48	Female	Southwest	Married	3	

```
      bmi_category smoking_status employment_status income_level  \
0           Normal     No Smoking          Salaried         <10L
1          Obesity        Regular          Salaried         <10L
2           Normal     No Smoking     Self-Employed    10L - 25L
3           Normal     No Smoking          Salaried        > 40L
4       Overweight        Regular     Self-Employed        > 40L
…              …              …                …          …
49903  Underweight     No Smoking     Self-Employed    25L - 40L
49904       Normal     No Smoking          Salaried        > 40L
49905       Normal        Regular        Freelancer    25L - 40L
49906       Normal     No Smoking          Salaried    10L - 25L
49907       Normal     Occasional     Self-Employed         <10L

       income_lakhs      medical_history insurance_plan  annual_premium_amount
0                 6              Diabetes         Bronze                   9053
1                 6              Diabetes         Bronze                  16339
2                20  High blood pressure         Silver                  18164
3                77           No Disease           Gold                  20303
4                99  High blood pressure         Silver                  13365
…               …                    …              …                      …
49903            35           No Disease         Bronze                   9111
49904            82              Thyroid           Gold                  27076
49905            32           No Disease         Bronze                   8564
49906            20           No Disease         Bronze                   9490
49907             7              Diabetes         Silver                  19730

[49908 rows x 13 columns]
```

[82]:
```
# After Treating outliers in 'income_lakhs' columns

sns.histplot(data=df2,x='income_lakhs')
plt.title('Distribution of Income (After)')
plt.xlabel('Income in Lakhs')
plt.ylabel('Number of people')
plt.show()
```



0.3 EDA

0.3.1 Univariate Analysis - Numeric Columns

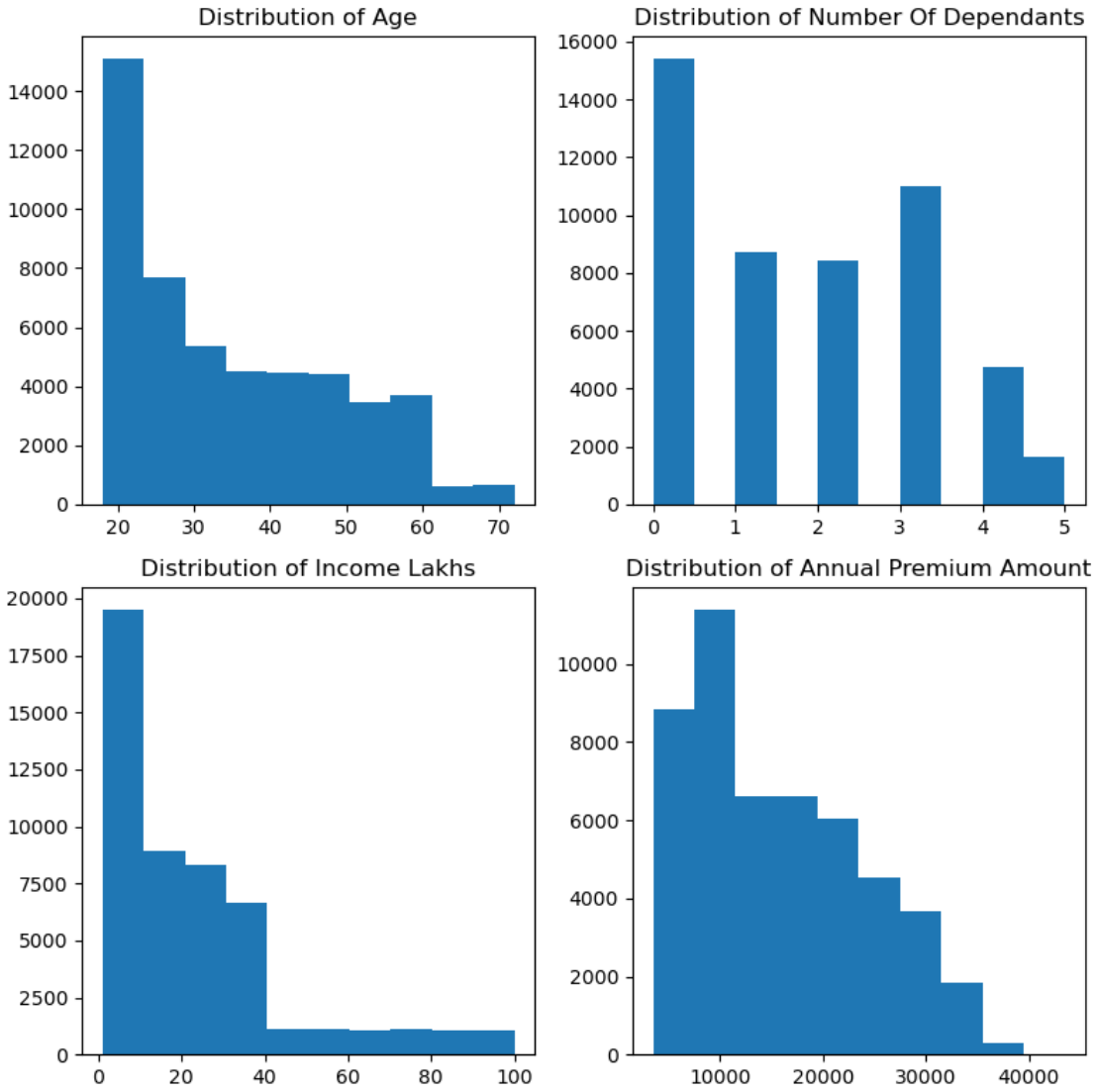
We will plot the distribution of numerical columns with Histogram

[83]:
```
numeric_columns
```

[83]:
```
Index(['age', 'number_of_dependants', 'income_lakhs', 'annual_premium_amount'],
dtype='object')
```

[84]:
```
# Plotting Distribution for all numerical columns

fig , ax = plt.subplots(2,2,figsize=(8,8))

# Iterator initiation to retrive one value at a time
it = iter(numeric_columns)

for i in range(2):
    for j in range(2):
        # To retrive one value at a time
```

```
        col = next(it)
        # Format column name for display: capitalize first letter of each word␣
 ↪and replace underscores with spaces
        x_ax = col.title().replace('_',' ')
        ax[i,j].hist(x=df2[col])
        ax[i,j].set_title(f'Distribution of {x_ax}')

# To adjust subplot parameters to give specified padding and prevent overlap of␣
 ↪labels
plt.tight_layout()
plt.show()
```



0.3.2 Univariate Analysis - Categorical Columns

We will plot the distribution of categorical columns with Histogram

[85]:
```
df2[cat_columns]
```

[85]:
```
       gender     region marital_status bmi_category smoking_status  \
0        Male  Northwest      Unmarried       Normal     No Smoking
1      Female  Southeast        Married      Obesity        Regular
2      Female  Northeast        Married       Normal     No Smoking
3      Female  Southeast        Married       Normal     No Smoking
4        Male  Northeast      Unmarried   Overweight        Regular
…         …        …              …           …              …
49903  Female  Northwest      Unmarried  Underweight     No Smoking
49904  Female  Southeast        Married       Normal     No Smoking
49905    Male  Northwest      Unmarried       Normal        Regular
49906    Male  Northwest      Unmarried       Normal     No Smoking
49907  Female  Southwest        Married       Normal     Occasional

      employment_status income_level      medical_history insurance_plan
0              Salaried         <10L             Diabetes         Bronze
1              Salaried         <10L             Diabetes         Bronze
2         Self-Employed    10L - 25L  High blood pressure         Silver
3              Salaried        > 40L           No Disease           Gold
4         Self-Employed        > 40L  High blood pressure         Silver
…                   …           …                   …            …
49903     Self-Employed    25L - 40L           No Disease         Bronze
49904          Salaried        > 40L              Thyroid           Gold
49905        Freelancer    25L - 40L           No Disease         Bronze
49906          Salaried    10L - 25L           No Disease         Bronze
49907     Self-Employed         <10L             Diabetes         Silver

[49908 rows x 9 columns]
```

In previous runs, the x-axis labels for the `medical_history` column were not aligning correctly in the plots. To resolve this issue, we will map the values using the dictionary ds_dict as shown below.

[86]:
```
ds_dict = {'No Disease': 'NO DISEASE',
 'Diabetes': 'DB',
 'High blood pressure': 'HBP',
 'Thyroid': 'THY',
 'Heart disease': 'HD',
 'Diabetes & High blood pressure': 'DB & HBP',
 'High blood pressure & Heart disease': 'HBP & HD',
 'Diabetes & Thyroid': 'DB & THY',
 'Diabetes & Heart disease': 'DB & HD'}

ds_dict
```

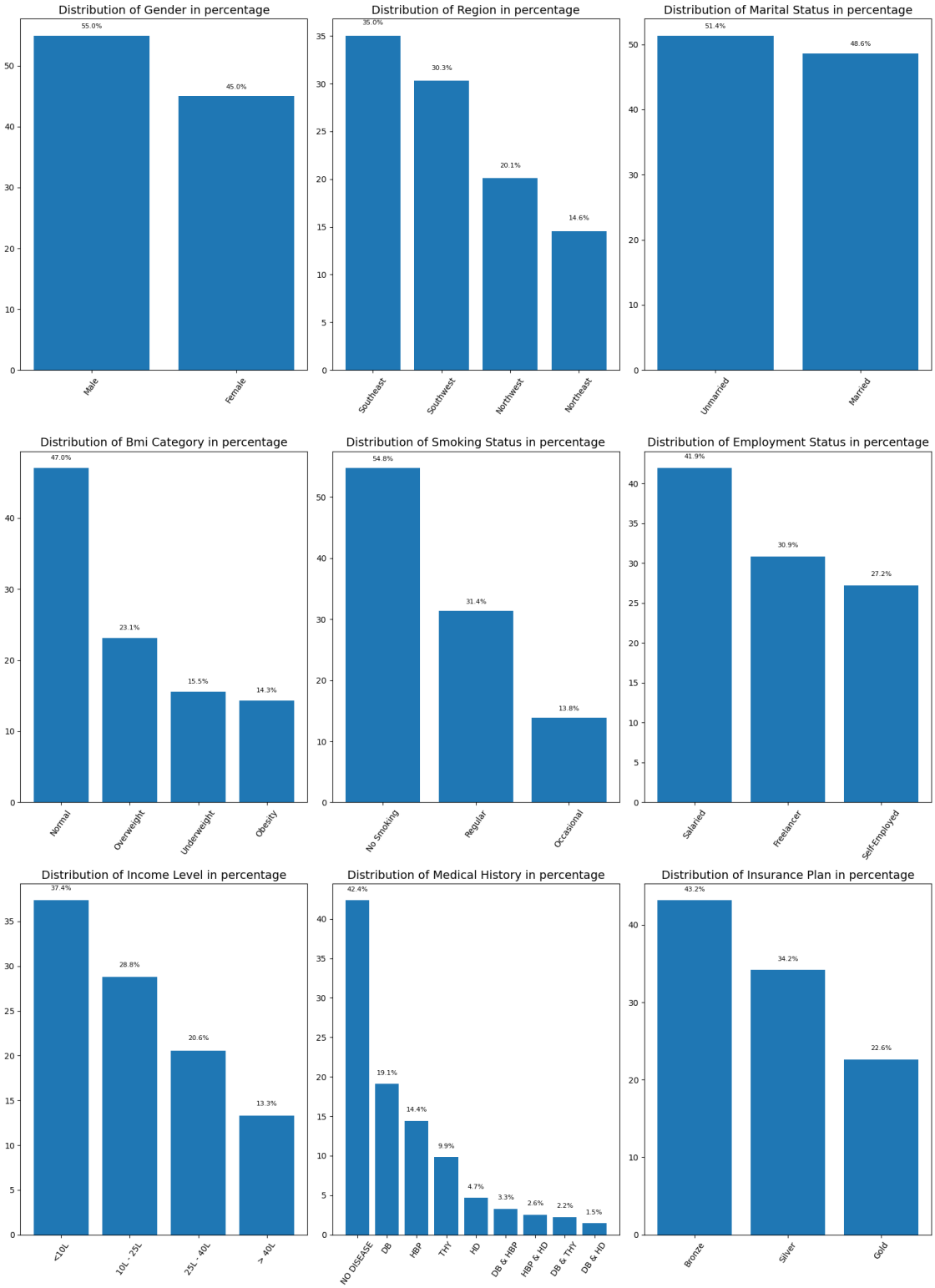
[86]:
```
{'No Disease': 'NO DISEASE',
 'Diabetes': 'DB',
 'High blood pressure': 'HBP',
 'Thyroid': 'THY',
 'Heart disease': 'HD',
 'Diabetes & High blood pressure': 'DB & HBP',
 'High blood pressure & Heart disease': 'HBP & HD',
 'Diabetes & Thyroid': 'DB & THY',
 'Diabetes & Heart disease': 'DB & HD'}
```

[87]:
```
# Showing how mapping works

print('Without Mapping:')
print(list(df2['medical_history'].value_counts(normalize=True).index))
print('\n','*'*100,'\n')
print('With Mapping:')
print(list(df2['medical_history'].value_counts(normalize=True).index.
 ↪map(ds_dict)))
```

```
Without Mapping:
['No Disease', 'Diabetes', 'High blood pressure', 'Thyroid', 'Heart disease',
'Diabetes & High blood pressure', 'High blood pressure & Heart disease',
'Diabetes & Thyroid', 'Diabetes & Heart disease']

 ****************************************************************************
**********************

With Mapping:
['NO DISEASE', 'DB', 'HBP', 'THY', 'HD', 'DB & HBP', 'HBP & HD', 'DB & THY', 'DB
& HD']
```

[88]:
```
# Plotting Distribution for all categorical columns

fig, ax = plt.subplots(3,3,figsize=(16,22))

# Iterator initiation to retrive one value at a time
it = iter(cat_columns)

for i in range(3):
    for j in range(3):
        # To retrive one value at a time
        col = next(it)

        # Format column name for display: capitalize first letter of each word␣
 ↪and replace underscores with spaces
        x_ax = col.title().replace('_',' ')
```

```
        # Map the 'medical_history' column using ds_dict to ensure consistent␣
 ↪and clean labels for plotting

        if col == 'medical_history':
            x_val = df2[col].value_counts(normalize=True).index.map(ds_dict)
        else:
            x_val = df2[col].value_counts(normalize=True).index

        h_val = df2[col].value_counts(normalize=True).values * 100
        ax[i,j].bar(x=x_val,height=h_val)
        ax[i,j].set_title(f'Distribution of {x_ax} in percentage',fontsize=14)
        ax[i,j].tick_params(axis='x', rotation=55)

        # Annotate each bar with its corresponding percentage value for better␣
 ↪interpretability
        for idx, val in enumerate(h_val):
            ax[i,j].text(idx, val + 1, f'{round(val,1)}%', ha='center',␣
 ↪va='bottom', fontsize=8)

plt.tight_layout()
plt.show()
```


Distribution of Gender in percentage
55.0%
45.0%
Male
Female
Distribution of Region in percentage
35.0%
30.3%
20.1%
14.6%
Southeast
Southwest
Northwest
Northeast
Distribution of Marital Status in percentage
51.4%
48.6%
Unmarried
Married
Distribution of Bmi Category in percentage
47.0%
23.1%
15.5%
14.3%
Normal
Overweight
Underweight
Obesity
Distribution of Smoking Status in percentage
54.8%
31.4%
13.8%
No Smoking
Regular
Occasional
Distribution of Employment Status in percentage
41.9%
30.9%
27.2%
Salaried
Freelancer
Self-Employed
Distribution of Income Level in percentage
37.4%
28.8%
20.6%
13.3%
<10L
10L - 25L
25L - 40L
> 40L
Distribution of Medical History in percentage
42.4%
19.1%
14.4%
9.9%
4.7%
3.3%
2.6%
2.2%
1.5%
NO DISEASE
DB
HBP
THY
HD
DB & HBP
HBP & HD
DB & THY
DB & HD
Distribution of Insurance Plan in percentage
43.2%
34.2%
22.6%
Bronze
Silver
Gold

0.3.3 Bivariate Analysis - Numeric Columns

We will plot the relationship between each numerical column (excluding the `annual_premium_amount` column itself) and the `annual_premium_amount` column to analyze potential correlations.

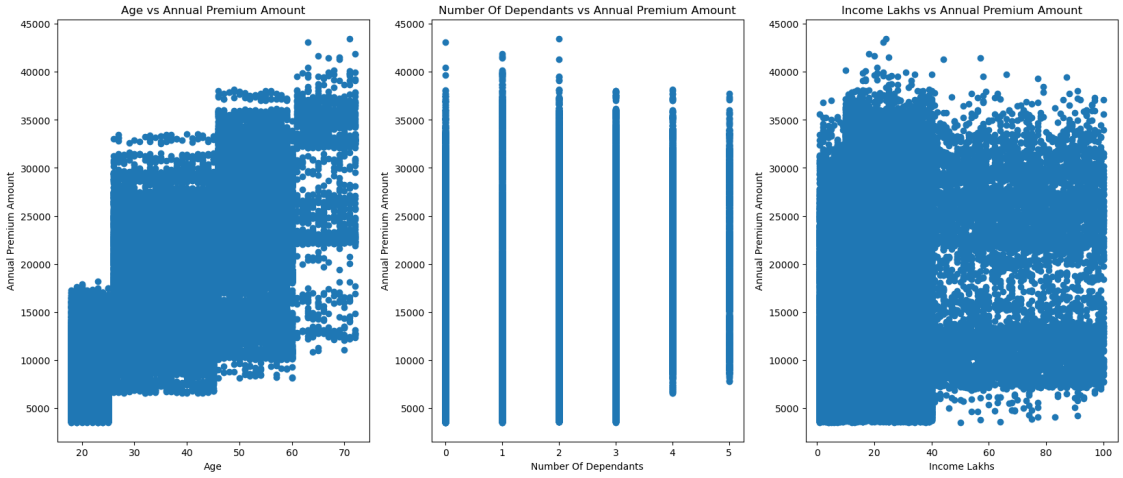
[89]:
```
df2[numeric_columns]
```

[89]:

	age	number_of_dependants	income_lakhs	annual_premium_amount
0	26	0	6	9053
1	29	2	6	16339
2	49	2	20	18164
3	30	3	77	20303
4	18	0	99	13365
…	…	…	…	…
49903	24	0	35	9111
49904	47	2	82	27076
49905	21	0	32	8564
49906	18	2	20	9490
49907	48	3	7	19730

[49908 rows x 4 columns]

[90]:
```
# Plotting the scatter plots

fig, ax = plt.subplots(1,3,figsize=(20,8))

for i in range(3):
    col = numeric_columns[i]

    # Format column names for display: capitalize first letter of each word and␣
 ↪replace underscores with spaces
    x_ax = col.title().replace('_',' ')
    y_ax = 'annual_premium_amount'.title().replace('_',' ')

    ax[i].scatter(x=df2[col],y=df2['annual_premium_amount'])
    ax[i].set_title(f'{x_ax} vs {y_ax}')
    ax[i].set_xlabel(x_ax)
    ax[i].set_ylabel(y_ax)
```



0.3.4 Bivariate Analysis - Categorical

We will explore the relationship between the categorical columns Income_Level and Insurance_Plan using multiple approaches:

- Cross-tabulation
- Grouped Bar Chart
- Stacked Bar Chart
- Heatmap

[91]:
```
df2.head()
```

[91]:

	age	gender	region	marital_status	number_of_dependants	bmi_category	smoking_status	employment_status	income_level	income_lakhs	medical_history	insurance_plan	annual_premium_amount
0	26	Male	Northwest	Unmarried	0	Normal	No Smoking	Salaried	<10L	6	Diabetes	Bronze	9053
1	29	Female	Southeast	Married	2	Obesity	Regular	Salaried	<10L	6	Diabetes	Bronze	16339
2	49	Female	Northeast	Married	2	Normal	No Smoking	Self-Employed	10L - 25L	20			
3	30	Female	Southeast	Married	3	Normal	No Smoking	Salaried	> 40L	77			
4	18	Male	Northeast	Unmarried	0	Overweight	Regular	Self-Employed	> 40L	99			

```
2  High blood pressure         Silver                 18164
3            No Disease           Gold                 20303
4  High blood pressure         Silver                 13365
```

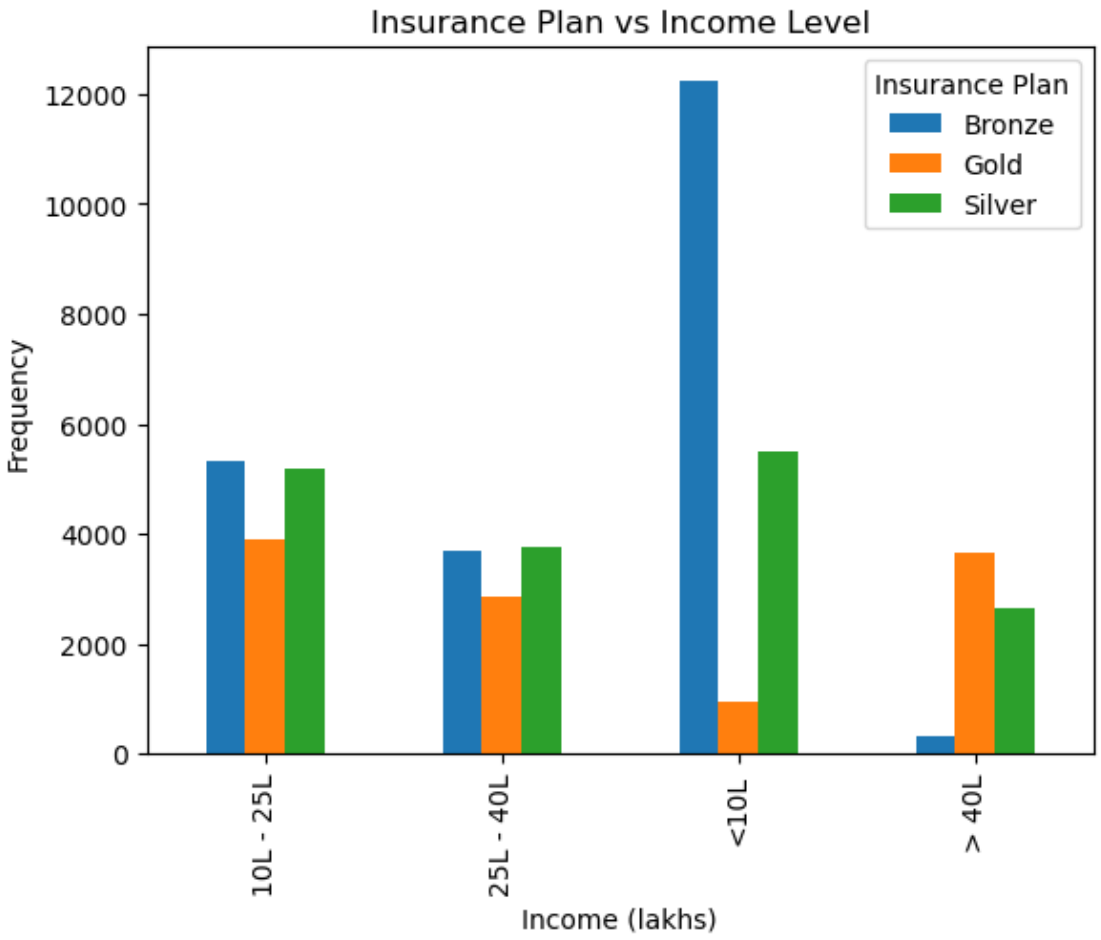
Cross-tabulation To examine the frequency distribution between `income_level` and `insurance_plan` using a cross-tabulation. This will show how many individuals fall into each combination of income level and insurance plan.

[92]:
```
ct = pd.crosstab(df['income_level'],df['insurance_plan'])
ct
```

[92]:
```
insurance_plan  Bronze  Gold  Silver
income_level
10L - 25L         5314  3886    5189
25L - 40L         3686  2844    3753
<10L             12239   931    5495
> 40L              330  3662    2647
```

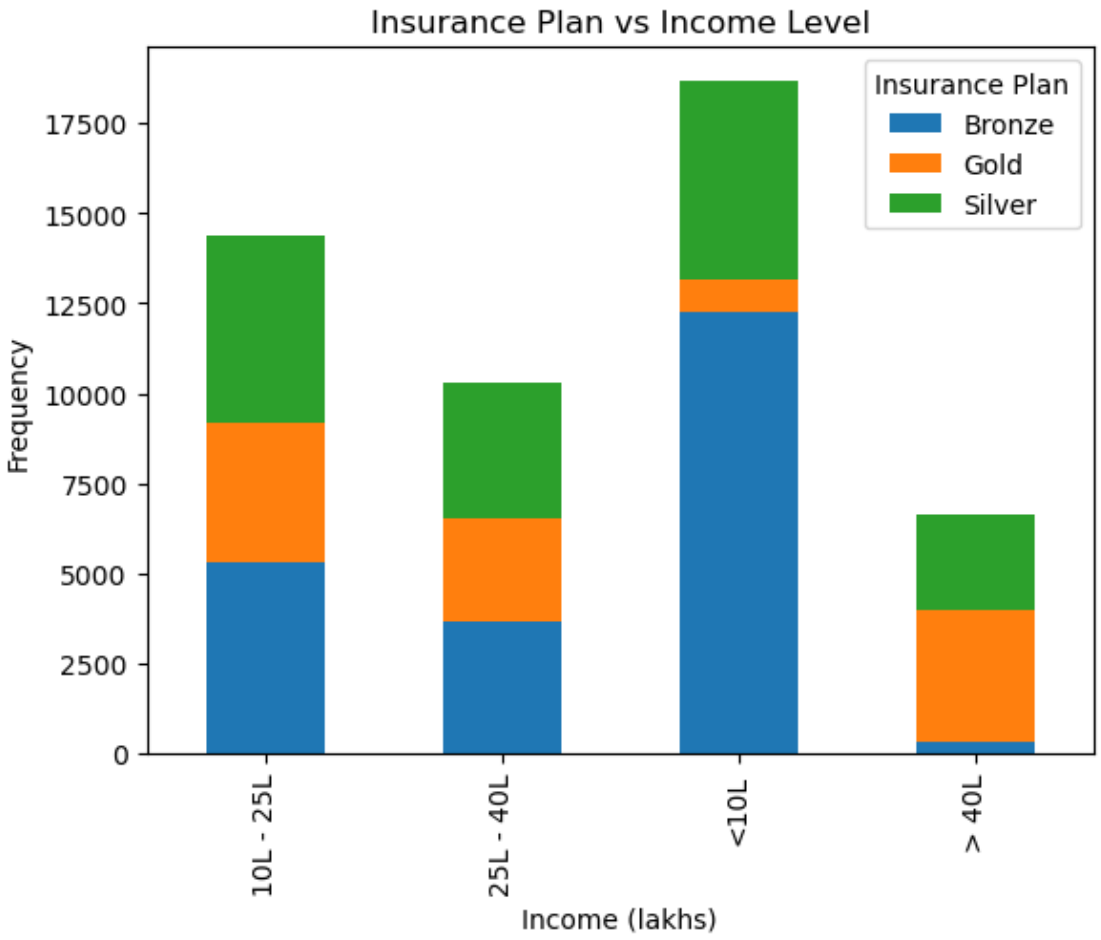
Grouped Bar Chart To visualize the count of each `insurance_plan` across different `income_level` categories. Each income level will have bars for the different insurance plans displayed side by side, allowing easy comparison.

[93]:
```
ct.plot(kind='bar')
plt.title('Insurance Plan vs Income Level')
plt.xlabel('Income (lakhs)')
plt.ylabel('Frequency')
plt.legend(title='Insurance Plan')
plt.show()
```



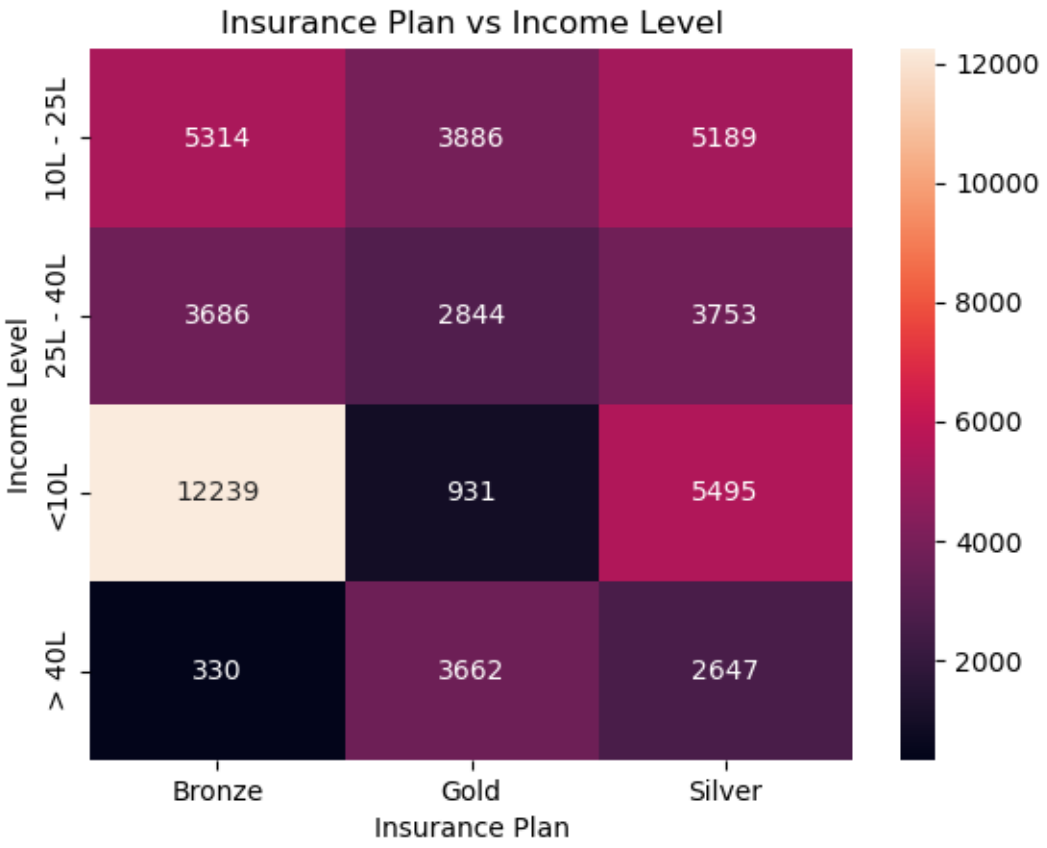
Stacked Bar Chart To Represent the distribution of `insurance_plan` within each `income_level` in a stacked format. This will help in understanding the proportion of each plan type within income categories.

[94]:
```
ct.plot(kind='bar',stacked=True)
plt.title('Insurance Plan vs Income Level')
plt.xlabel('Income (lakhs)')
plt.ylabel('Frequency')
plt.legend(title='Insurance Plan')
plt.show()
```



Heatmap To visualize the intensity of the relationship between `insurance_plan` and `income_level`. The cells will be color-coded based on frequency, highlighting patterns and concentrations in the data.

[95]:
```
sns.heatmap(ct,annot=True,fmt='0')
plt.title('Insurance Plan vs Income Level')
plt.xlabel('Insurance Plan')
plt.ylabel('Income Level')
plt.show()
```


Insurance Plan vs Income Level
10L - 25L
25L - 40L
<10L
> 40L
Income Level
5314
3886
5189
3686
2844
3753
12239
931
5495
330
3662
2647
Bronze
Gold
Silver
Insurance Plan
12000
10000
8000
6000
4000
2000