week2_premium_estimator_main

June 23, 2025

[5]:
```
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

[6]:
```
# 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
```

[7]:
```
# Reading the data

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

[8]:
```
# Displaying the data

df.head()
```

[8]:
```
   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
```

[9]:
```python
df.shape
```

[9]:
```
(50000, 13)
```

[10]:
```python
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

The following steps were performed during data cleaning:

- Handling missing values

- Removing duplicated rows
- Formatting numeric and categorical values
- Treating outliers to improve data quality

0.2.1 Column Formatter

[11]:
```python
# Renaming the columns with proper formatter
# Eg: Number Of Dependants -> number_of_dependants

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

[12]:
```python
df.head()
```

[12]:
```
   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

[13]:
```python
# Finding the columns with NA values

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

[13]:
```
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
```

[14]:
```
# Dropping the Rows where it has NA value

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

[14]:
```
       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
```

Since the count is low, the rows containing at least one missing value were dropped.

[15]:
```
# Dropping
df.dropna(how='any',inplace = True)
```

[16]:
```
# After Dropping
df[df['smoking_status'].isna()]
```

[16]:
```
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: []
```

[17]:
```
df.isna().sum()
```

[17]:
```
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
```

[18]:
```
df.reset_index(inplace=True,drop=True)
```

0.2.3 Handling Duplicated Rows

[19]:
```
# Duplicated rows

df[df.duplicated()]
```

[19]:
```
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: []
```

Although there are no duplicated rows, they are being dropped as a precautionary measure.

[20]:
```
# Dropping the duplicated rows

df.drop_duplicates(inplace=True)
```

0.2.4 Fomatting Values - Numeric Columns

[21]:
```
df.dtypes
```

[21]:
```
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
```

[22]:
```
# Selecting only the columns with numerical values

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

[22]:
```
Index(['age', 'number_of_dependants', 'income_lakhs', 'annual_premium_amount'],
dtype='object')
```

[23]:
```
# 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]
****

The negative values in `number_of_dependents` should be handled by converting them to their absolute values.

[24]:
```
# Before formatting

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

[24]:
```
array([ 0,  2,  3,  4,  1,  5, -3, -1])
```

[25]:
```
df['number_of_dependants'] = abs(df['number_of_dependants'])
```

[26]:
```
# After formatting

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

[26]:
```
array([0, 2, 3, 4, 1, 5])
```

0.2.5 Fomatting Values - Categorical Columns

[27]:
```
# Selecting only the columns with categorical values

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

[27]:
```
Index(['gender', 'region', 'marital_status', 'bmi_category', 'smoking_status',
       'employment_status', 'income_level', 'medical_history',
       'insurance_plan'],
      dtype='object')
```

[28]:
```
# 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 indicated by the above result, the `smoking_status` column contains multiple values with the same meaning; therefore, this column will be formatted.

[29]:
```python
# Before formatting

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

[29]:
```
array(['No Smoking', 'Regular', 'Occasional', 'Smoking=0',
       'Does Not Smoke', 'Not Smoking'], dtype=object)
```

[30]:
```python
# 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',
    }
)
```

[31]:
```
# After formatting

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

[31]:
```
array(['No Smoking', 'Regular', 'Occasional'], dtype=object)
```

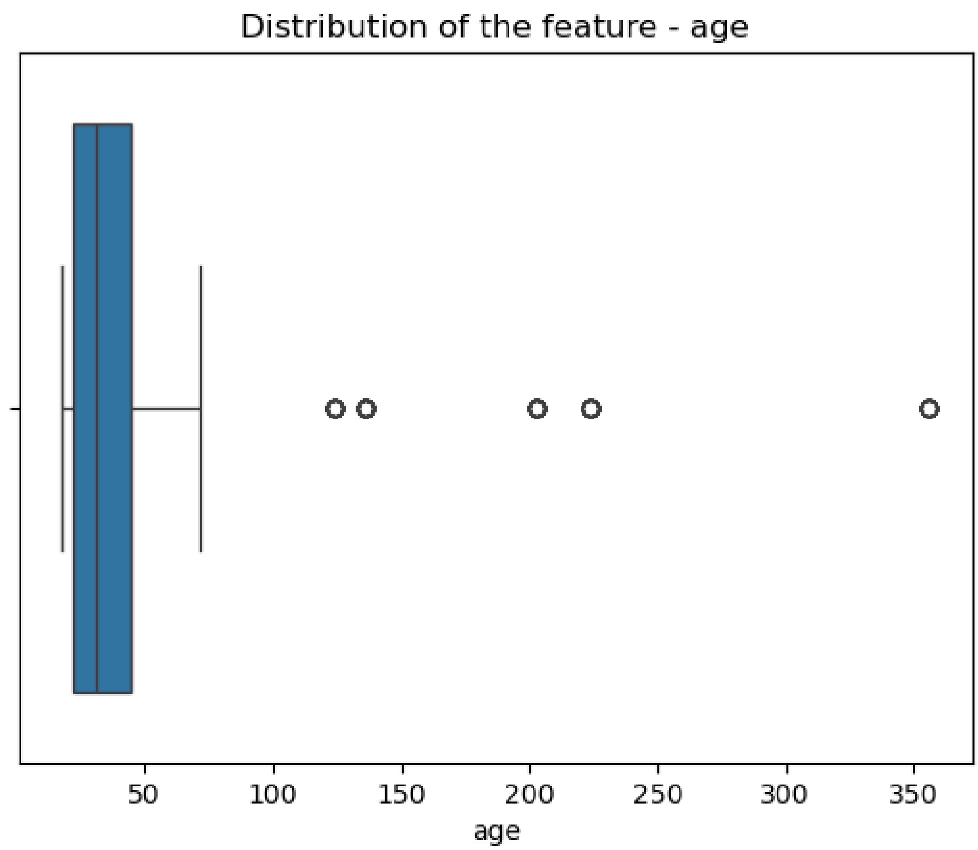
0.2.6 Outlier Treatment

[32]:
```
df.describe()
```

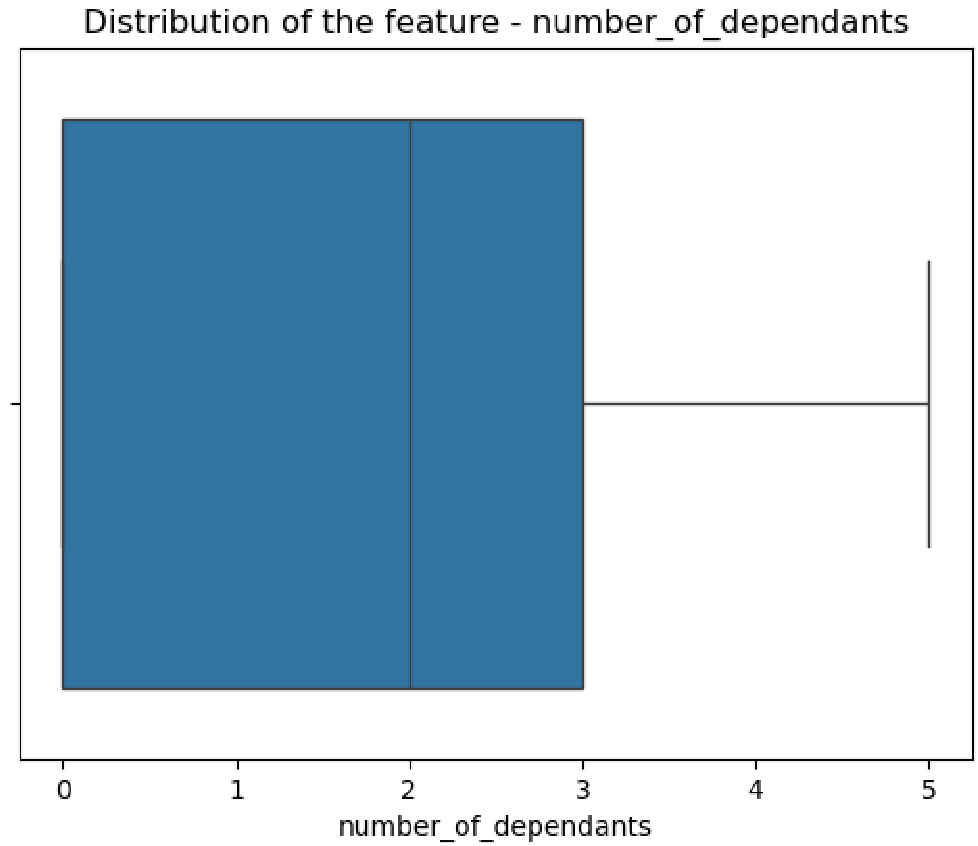
[32]:

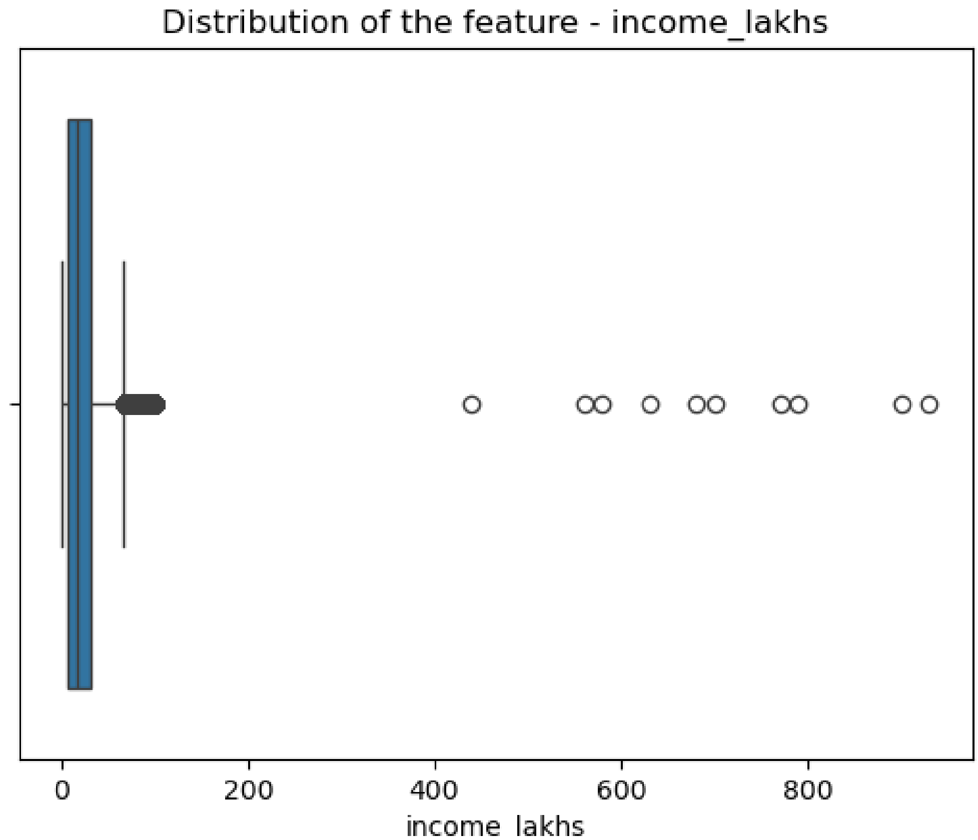
	age	number_of_dependants	income_lakhs	annual_premium_amount
count	49976.000000	49976.000000	49976.000000	49976.000000
mean	34.591764	1.717284	23.021150	15766.810189
std	15.000378	1.491953	24.221794	8419.995271
min	18.000000	0.000000	1.000000	3501.000000
25%	22.000000	0.000000	7.000000	8607.750000
50%	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

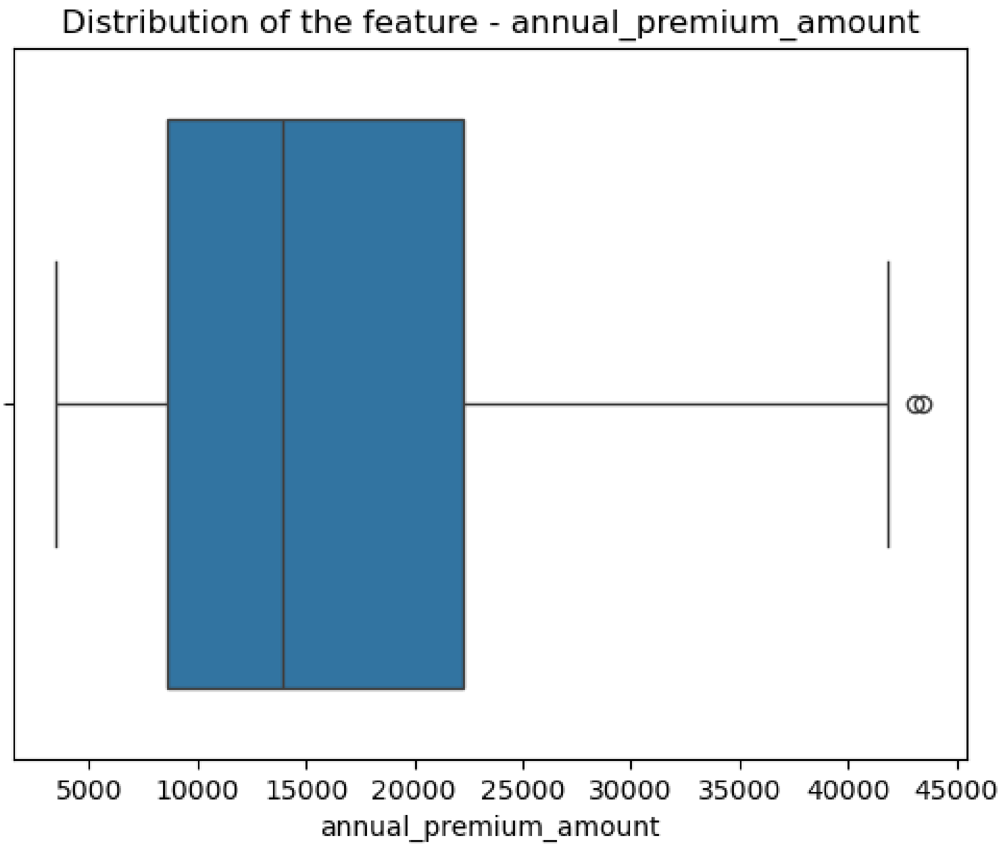
[33]:
```
# Plotting box plot for each numerical columns to detect outliers

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 observed above, outliers are present in the `age` and `income_lakhs` columns. This is also illustrated by the box plot shown above.

Age

[34]:
```python
# Before
df['age'].unique()
```

[34]:
```
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])
```

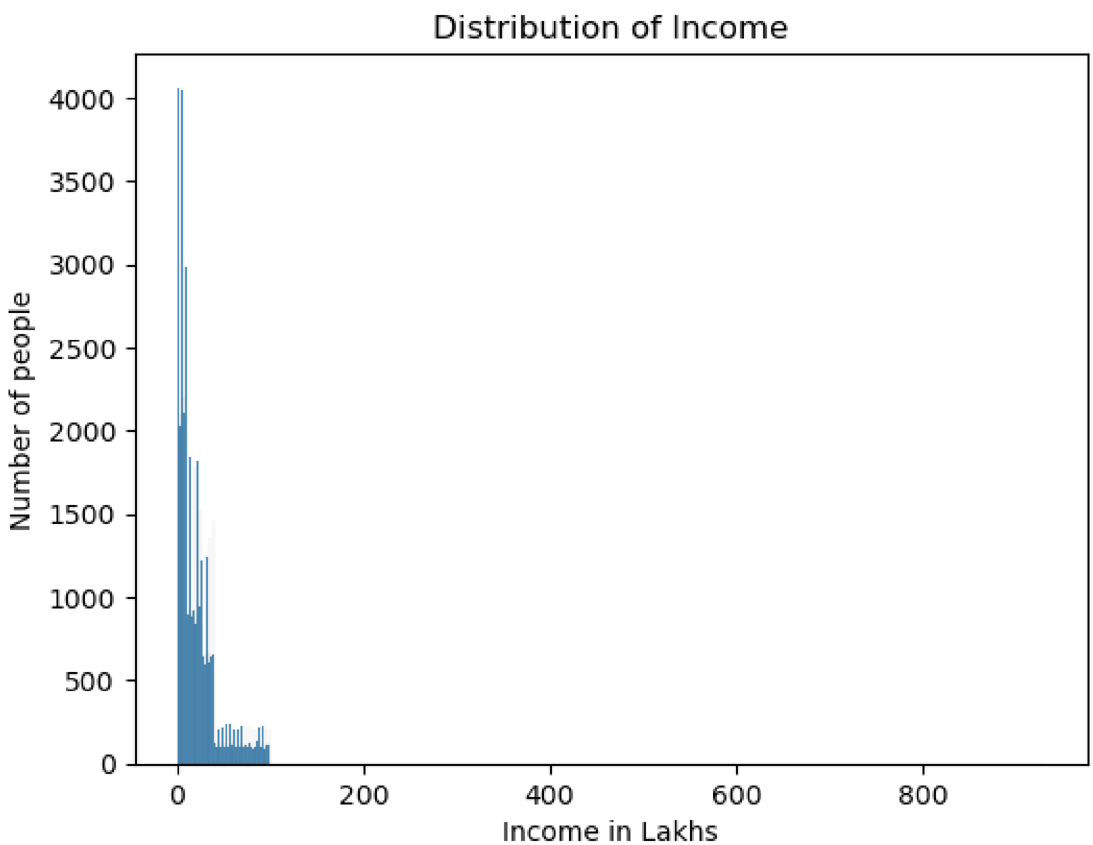
[35]:
```python
# Selecting only the rows where age <= 100. Because age > 100 is outlier

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

[36]:
```python
# After
df1['age'].unique()
```

[36]:
```
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

[37]:
```python
# 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()
```



[38]:
```python
# 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:

The current upper boundary is too low. Therefore, consultation with the business team will be conducted to determine the optimal upper boundary for identifying outliers in the income_lakhs column.

It has been decided that the upper boundary will be set at 1 crore (i.e., 100 lakhs). Any values exceeding this threshold will be considered outliers.

[39]:
```
# Selecting only the rows where income <= 100 lakhs. Because income > 100 lakhs
↪is outlier

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

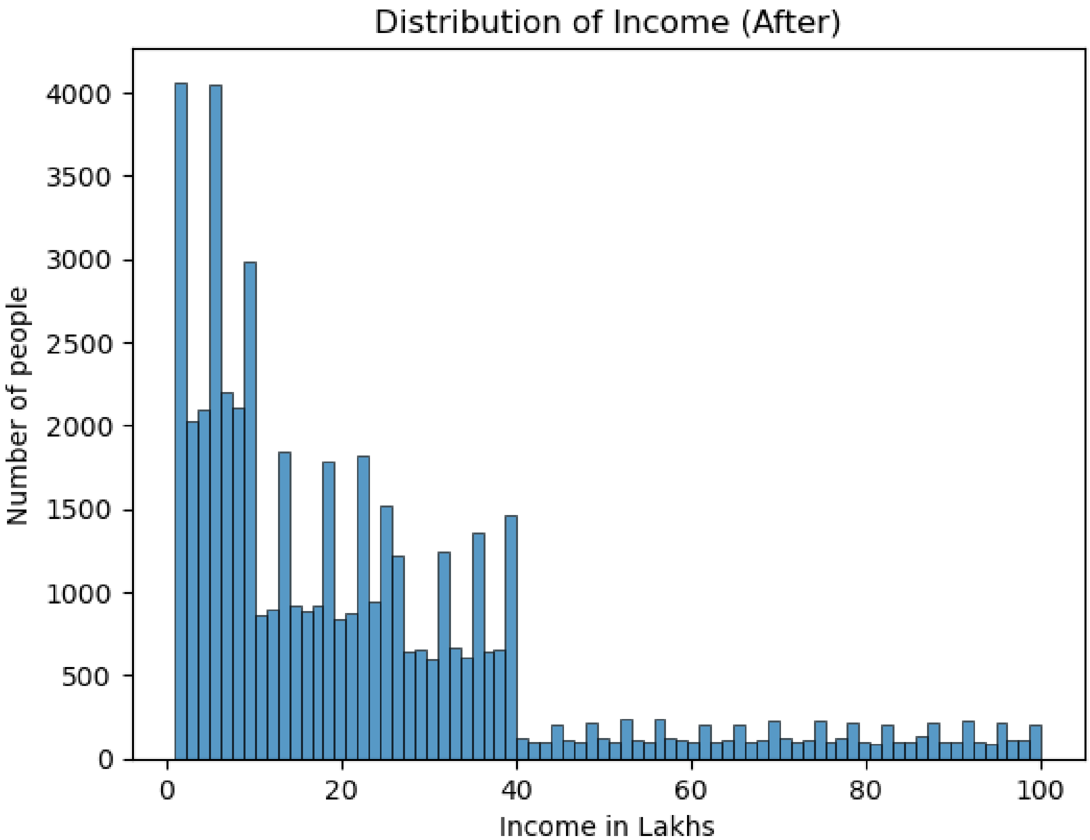
[40]:
```
df2.reset_index(drop=True,inplace=True)
```

[41]:
```
df2
```

[41]:
```
       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]
```

[42]:
```python
# 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

Exploratory Data Analysis (EDA) is performed as follows:

- Univariate analysis on numeric columns
- Univariate analysis on categorical columns
- Bivariate analysis on numeric columns
- Bivariate analysis on categorical columns

0.3.1 Univariate Analysis - Numeric Columns

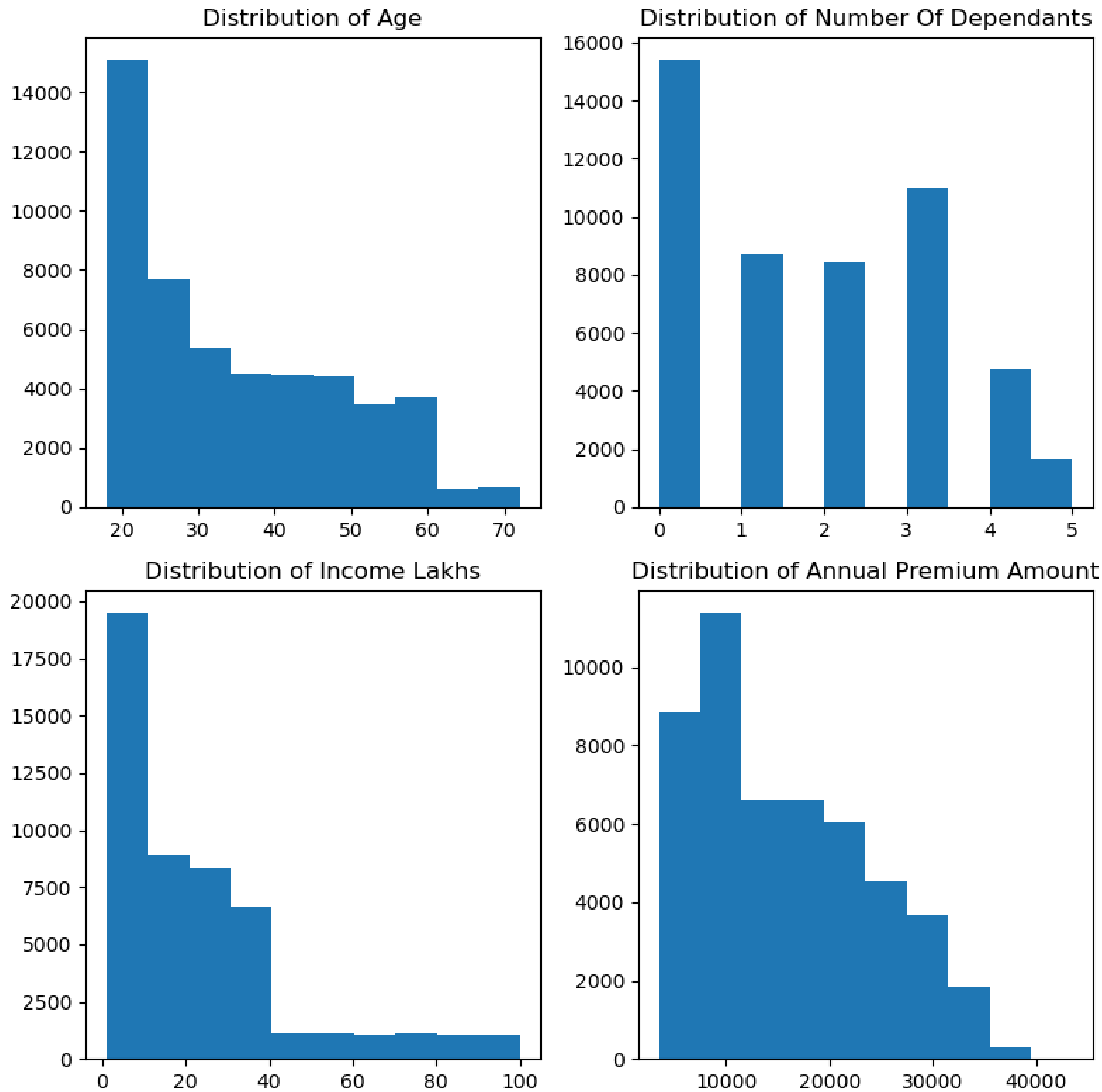
The distribution of numerical columns will be plotted using histograms.

[43]:
```
numeric_columns
```

[43]:
```
Index(['age', 'number_of_dependants', 'income_lakhs', 'annual_premium_amount'],
dtype='object')
```

[44]:
```
# 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

The distribution of categorical columns will be plotted using barcharts.

[45]:
```
df2[cat_columns]
```

[45]:
```
       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 misaligned in the plots. To resolve this issue, the values will be mapped using the dictionary *ds_dict* as shown below.

[46]:
```
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
```

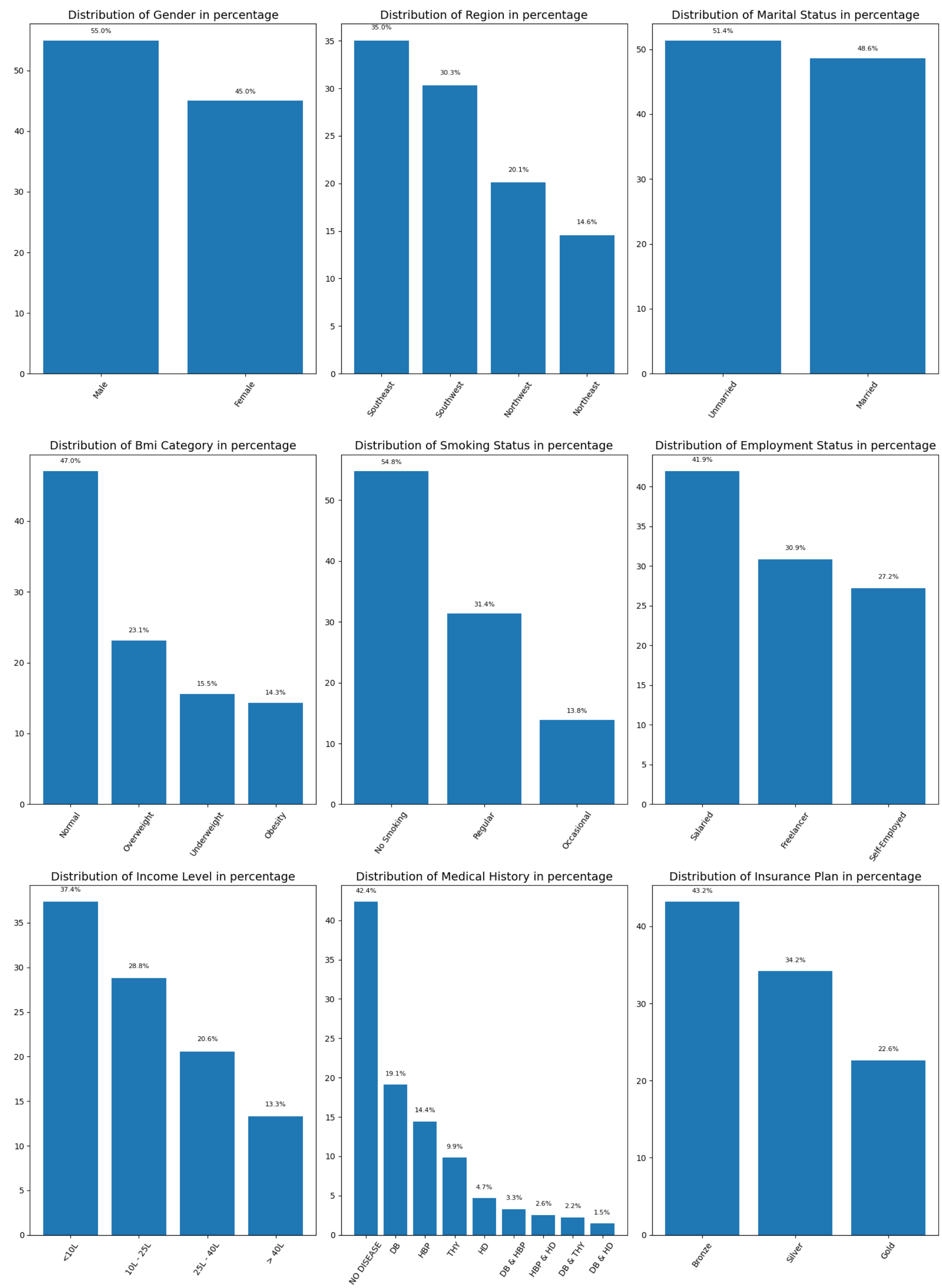
[46]:
```
{'No Disease': 'NO DISEASE',
 'Diabetes': 'DB',
 'High blood pressure': 'HBP',
 'Thyroid': 'THY',
 'Heart disease': 'HD',
 'Diabetes & High blood pressure': 'DB & HBP',
 'High blood pressure & Heart disease': 'HBP & HD',
 'Diabetes & Thyroid': 'DB & THY',
 'Diabetes & Heart disease': 'DB & HD'}
```

[47]:
```
# 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']
```

[48]:
```
# 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

The relationship between each numerical column (excluding the `annual_premium_amount` column itself) and the `annual_premium_amount` column will be plotted to analyze potential correlations.

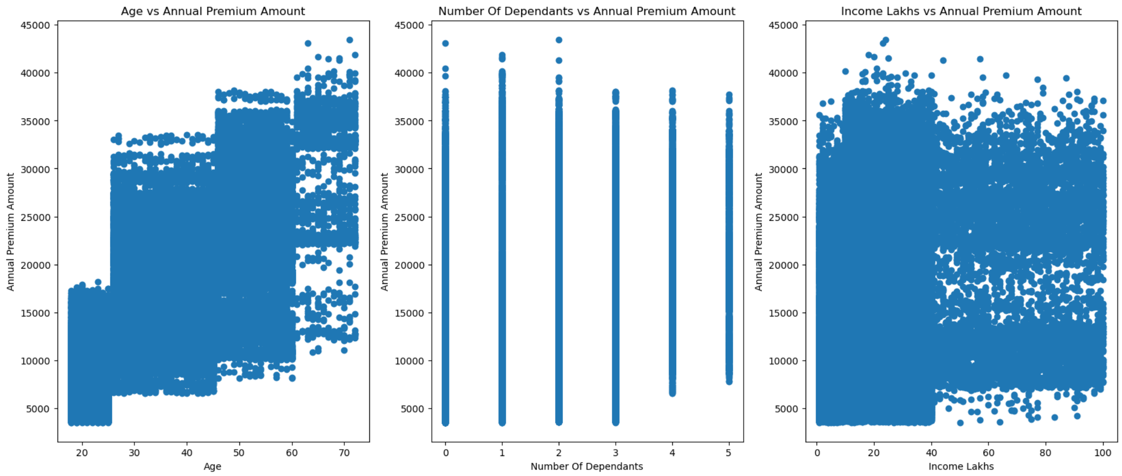
[49]:
```
df2[numeric_columns]
```

[49]:

	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]

[50]:
```
# 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

The relationship between the categorical columns `income_level` and `insurance_plan` will be explored using multiple approaches:

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

[51]:
```python
df2.head()
```

[51]:
```
   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
```

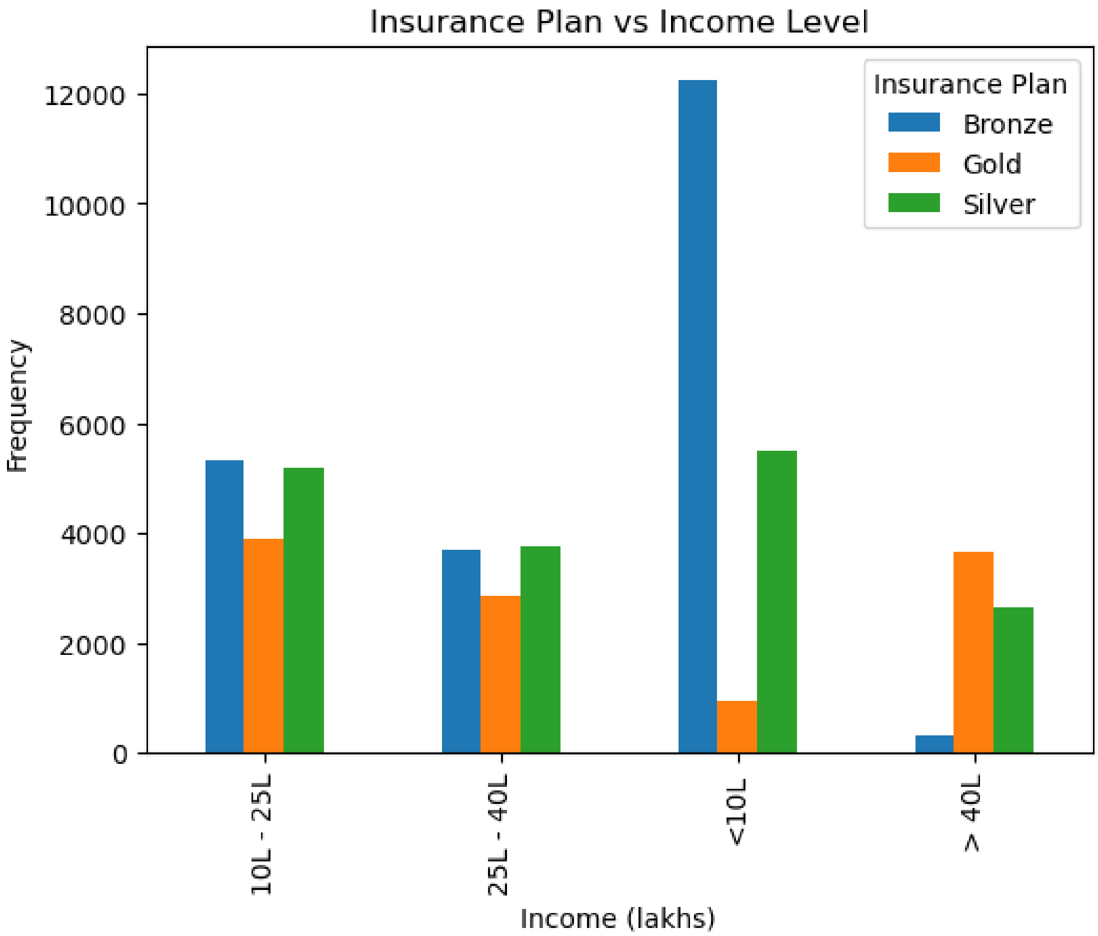
Cross-tabulation The frequency distribution between `income_level` and `insurance_plan` will be examined using a cross-tabulation, which will display the number of individuals in each combination of income level and insurance plan.

[52]:
```
ct = pd.crosstab(df['income_level'],df['insurance_plan'])
ct
```

[52]:
```
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
```

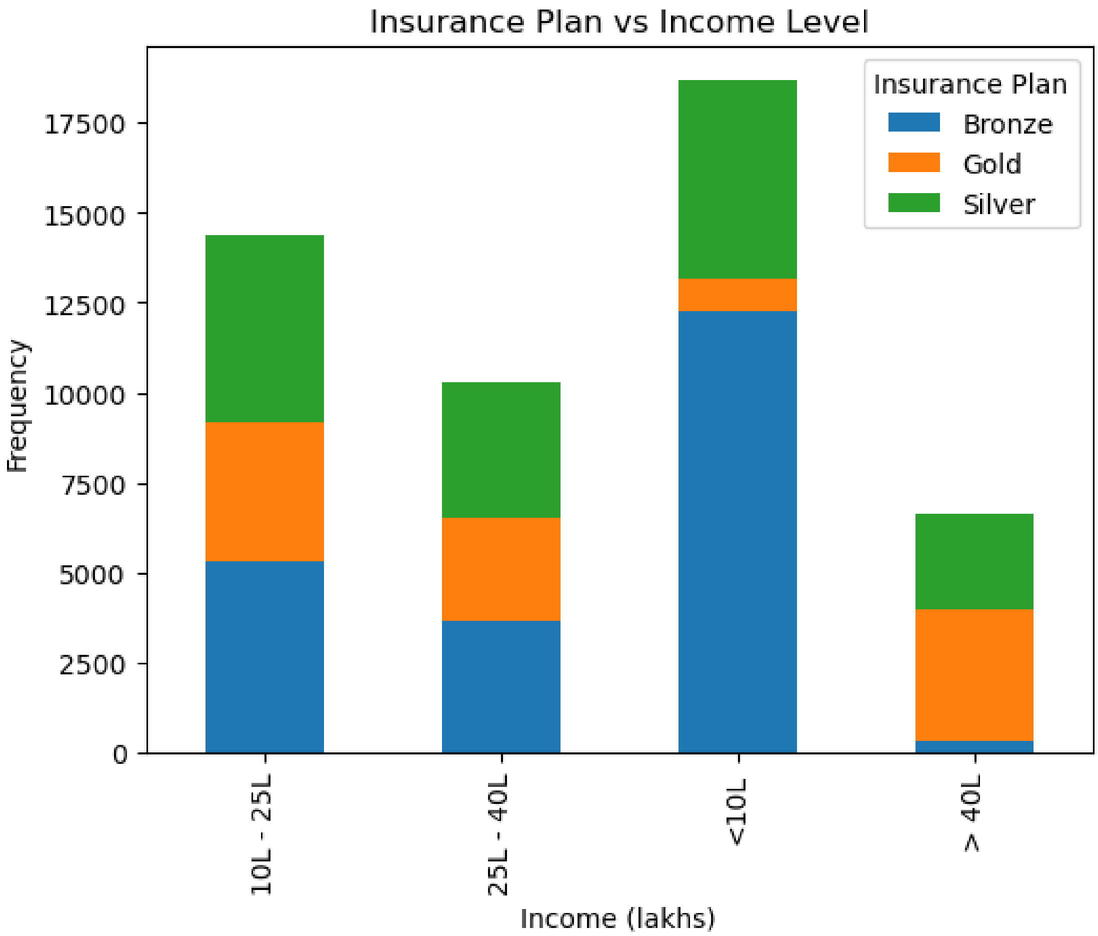
Grouped Bar Chart The count of each `insurance_plan` across different `income_level` categories will be visualized using grouped bar charts, where each income level will have side-by-side bars representing the different insurance plans for easy comparison.

[53]:
```
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()
```



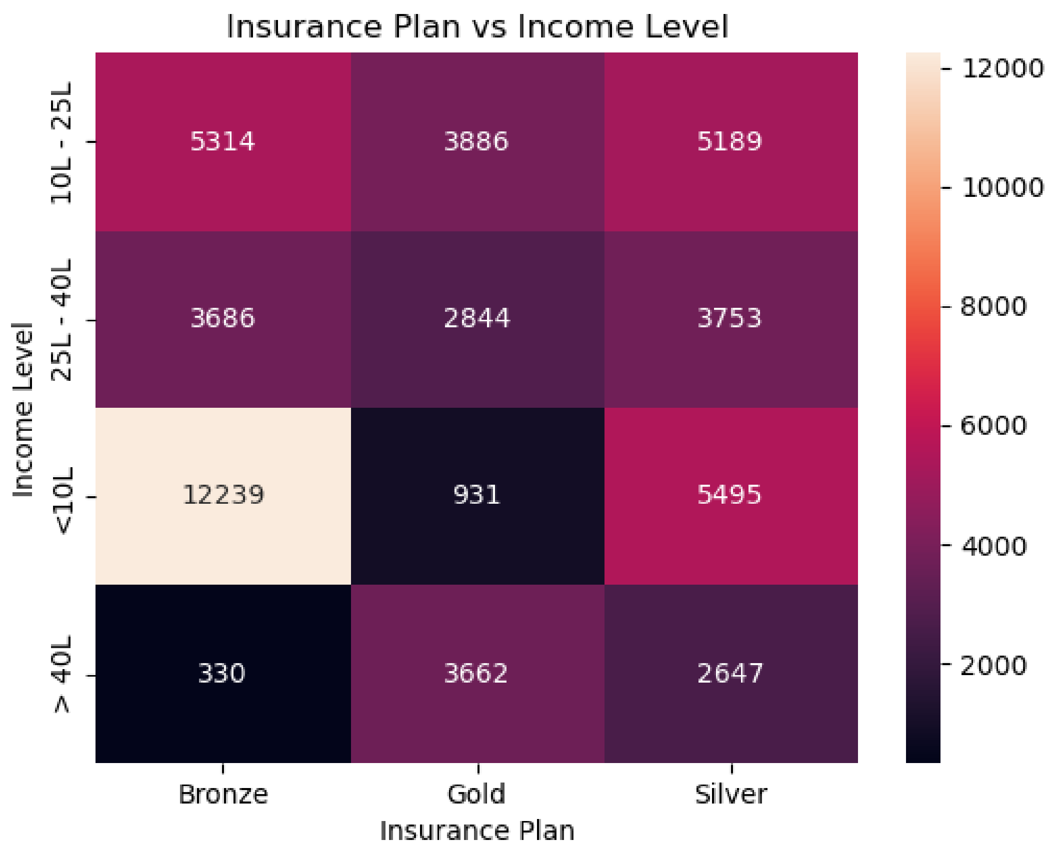
Stacked Bar Chart The distribution of `insurance_plan` within each `income_level` will be represented in a stacked bar chart format, helping to understand the proportion of each plan type within the income categories.

[54]:
```python
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 The intensity of the relationship between `insurance_plan` and `income_level` will be visualized using a heatmap, where the cells are color-coded based on frequency to highlight patterns and concentrations in the data.

[55]:
```
sns.heatmap(ct,annot=True,fmt='0')
plt.title('Insurance Plan vs Income Level')
plt.xlabel('Insurance Plan')
plt.ylabel('Income Level')
plt.show()
```



0.4 Feature Engineering

The following feature engineering steps were performed:

- Created new features as required
- Transformed features, including encoding categorical variables
- Selected important features using statistical methods such as Variance Inflation Factor (VIF) and correlation analysis

0.4.1 Feature Creation - 'total_risk_score'

In this we created a new feature `total_risk_score` from `medical_history` column

[56]:
```
# Extract all distinct medical conditions listed in the dataset

df2['medical_history'].unique()
```

[56]:
```
array(['Diabetes', 'High blood pressure', 'No Disease',
       'Diabetes & High blood pressure', 'Thyroid', 'Heart disease',
       'High blood pressure & Heart disease', 'Diabetes & Thyroid',
       'Diabetes & Heart disease'], dtype=object)
```

The following risk scores, as provided by the business, will be assigned to the corresponding medical conditions:

- Heart disease: 8
- Diabetes: 6
- High blood pressure: 6
- Thyroid: 5
- No Disease: 0
- None: 0

[57]:
```
df3 = df2.copy()
df3.head()
```

[57]:
```
   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
```

[58]:
```
# Split the 'medical_history' column into 'disease1' and 'disease2' using '&'␣
↪as the delimiter

df3[['disease1','disease2']] = df3['medical_history'].str.lower().str.split(' &␣
↪',expand=True)
df3.head()
```

[58]:
```
   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

              disease1 disease2
0             diabetes     None
1             diabetes     None
2  high blood pressure     None
3           no disease     None
4  high blood pressure     None
```

[59]:
```python
# Risk Score Dictionary

risk_score_dict = {
    'heart disease' : 8,
'diabetes' : 6,
'high blood pressure' : 6,
'thyroid' : 5,
'no disease' : 0,
None : 0
}
```

[60]:
```python
# Mapping each disease to its corresponding score using a predefined dictionary

df3['disease1_score'] = df3['disease1'].map(risk_score_dict)
df3['disease2_score'] = df3['disease2'].map(risk_score_dict)
df3.sample(2)
```

[60]:
```
       age gender    region marital_status  number_of_dependants  \
11647   53   Male  Southeast        Married                     3
5418    59   Male  Southeast        Married                     2

      bmi_category smoking_status employment_status income_level  \
11647      Obesity     Occasional          Salaried         <10L
5418        Normal        Regular     Self-Employed         <10L

       income_lakhs medical_history insurance_plan  annual_premium_amount  \
```

```
11647              3         Thyroid         Bronze                       15240
5418               3         Thyroid         Bronze                       14013

      disease1 disease2  disease1_score  disease2_score
11647  thyroid     None               5               0
5418   thyroid     None               5               0
```

[61]:
```
# Check if all diseases have been assigned a score and identify any missing
  ↪values

print('Unique Scores in Disease1 -> ',df3['disease1_score'].unique())
print('Unique Scores in Disease2 -> ',df3['disease2_score'].unique())
```

```
Unique Scores in Disease1 ->  [6 0 5 8]
Unique Scores in Disease2 ->  [0 6 8 5]
```

Since there are no NaN values, it can be concluded that all entries in the `disease1` and `disease2` columns have been successfully mapped.

[62]:
```
# Calculating the total risk score by summing 'disease1_score' and
  ↪'disease2_score'

df3['total_risk_score'] = df3['disease1_score'] + df3['disease2_score']
df3.sample(4)
```

[62]:
```
       age gender     region marital_status  number_of_dependants  \
2358    69   Male  Southeast        Married                     1
1268    60   Male  Southwest        Married                     3
9998    22   Male  Southwest        Married                     3
27335   56   Male  Southwest        Married                     5

      bmi_category smoking_status employment_status income_level  \
2358        Normal     No Smoking     Self-Employed    25L - 40L
1268        Normal     No Smoking     Self-Employed        > 40L
9998        Normal     No Smoking        Freelancer         <10L
27335       Normal        Regular          Salaried         <10L

       income_lakhs      medical_history insurance_plan  \
2358             28              Thyroid         Bronze
1268             45  High blood pressure           Gold
9998              7             Diabetes         Bronze
27335             8  High blood pressure         Silver

       annual_premium_amount             disease1 disease2  disease1_score  \
2358                   12709              thyroid     None               5
1268                   26541  high blood pressure     None               6
9998                    9001             diabetes     None               6
27335                  21654  high blood pressure     None               6
```

```
       disease2_score  total_risk_score
2358                0                 5
1268                0                 6
9998                0                 6
27335               0                 6
```

The following columns are being dropped as they are no longer needed: `medical_history`, `disease1`, `disease2`, `disease1_score`, and `disease2_score`.

[63]:
```
cols_to_drop = ['medical_history', 'disease1', 'disease2', 'disease1_score',␣
 ↪'disease2_score']
df4 = df3.drop(cols_to_drop,axis=1)
df4.reset_index(drop=True,inplace=True)
df4
```

[63]:
```
       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 insurance_plan  annual_premium_amount  total_risk_score
0                 6         Bronze                   9053                 6
1                 6         Bronze                  16339                 6
2                20         Silver                  18164                 6
3                77           Gold                  20303                 0
4                99         Silver                  13365                 6
```

```
...              ...        ...                         ...            ...
49903             35     Bronze                        9111              0
49904             82       Gold                       27076              5
49905             32     Bronze                        8564              0
49906             20     Bronze                        9490              0
49907              7     Silver                       19730              6

[49908 rows x 13 columns]
```

0.4.2 Feature Cleaning & Transformation

The following transformations were applied to the dataset features:

- Label Encoding for ordinal categorical variables (to preserve order)
- One Hot Encodingfor nominal categorical variables (to avoid introducing ordinal relationships)

[64]:
```python
df5 = df4.copy()
df5.head()
```

[64]:
```
   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 insurance_plan  \
0     No Smoking          Salaried         <10L             6         Bronze
1        Regular          Salaried         <10L             6         Bronze
2     No Smoking     Self-Employed    10L - 25L            20         Silver
3     No Smoking          Salaried        > 40L            77           Gold
4        Regular     Self-Employed        > 40L            99         Silver

   annual_premium_amount  total_risk_score
0                   9053                 6
1                  16339                 6
2                  18164                 6
3                  20303                 0
4                  13365                 6
```

Label Encodig - 'income_level'

[65]:
```python
# Extract all distinct income levels listed in the dataset

df5.income_level.unique()
```

[65]:
```
array(['<10L', '10L - 25L', '> 40L', '25L - 40L'], dtype=object)
```

```
[66]: # Income level dictionary

income_level_dict =  {
        '<10L' : 1,
        '10L - 25L' : 2,
        '25L - 40L' : 3,
        '> 40L' : 4
    }
```

```
[67]: # Mapping each income level to a value using a predefined dictionary

df5['income_level'] = df5['income_level'].map(income_level_dict)
```

```
[68]: # After mapping

df5.income_level.unique()
```

```
[68]: array([1, 2, 4, 3])
```

Label Encodig - 'insurance_plan'

```
[69]: # Extract all distinct insurance plan listed in the dataset

df5.insurance_plan.unique()
```

```
[69]: array(['Bronze', 'Silver', 'Gold'], dtype=object)
```

```
[70]: # Insurance Plan dictionary

insurance_plan_dict = {
        'Bronze' : 1,
        'Silver' : 2,
        'Gold' : 3,
    }
```

```
[71]: # Mapping each insurance plan to a value using a predefined dictionary

df5['insurance_plan'] = df5['insurance_plan'].map(insurance_plan_dict)
```

```
[72]: # After mapping

df5.insurance_plan.unique()
```

```
[72]: array([1, 2, 3])
```

One Hot Encoding

```
[73]: # Selecting columns to perform one hot encoding
```

```
cols_to_encode = ['gender', 'region', 'marital_status', 'bmi_category',␣
↪'smoking_status', 'employment_status']
cols_to_encode
```

[73]:
```
['gender',
 'region',
 'marital_status',
 'bmi_category',
 'smoking_status',
 'employment_status']
```

[74]:
```
# Performing One Hot Encoding on df5

df6 = pd.get_dummies(df5,columns = cols_to_encode,dtype=int,drop_first=True)
```

[75]:
```
df6.sample(5)
```

[75]:
```
       age  number_of_dependants  income_level  income_lakhs  insurance_plan  \
41672   41                     1             4            72               3
1380    20                     0             3            27               1
9953    18                     2             1             7               1
7862    22                     2             3            31               1
16612   53                     2             2            10               3

       annual_premium_amount  total_risk_score  gender_Male  region_Northwest  \
41672                  21674                 0            0                 0
1380                    5773                 0            1                 0
9953                    9530                 0            1                 0
7862                    9690                 0            0                 0
16612                  29848                 6            1                 0

       region_Southeast  region_Southwest  marital_status_Unmarried  \
41672                 0                 1                         1
1380                  0                 1                         1
9953                  1                 0                         1
7862                  1                 0                         0
16612                 1                 0                         0

       bmi_category_Obesity  bmi_category_Overweight  \
41672                     0                        0
1380                      0                        0
9953                      0                        0
7862                      0                        1
16612                     0                        0

       bmi_category_Underweight  smoking_status_Occasional  \
41672                         1                          1
```

```
1380                         1                          0
9953                         1                          0
7862                         0                          0
16612                        0                          0

       smoking_status_Regular  employment_status_Salaried  \
41672                       0                           0
1380                        1                           1
9953                        0                           1
7862                        0                           0
16612                       1                           0

       employment_status_Self-Employed
41672                                1
1380                                 0
9953                                 0
7862                                 1
16612                                1
```

[76]:
```
df6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49908 entries, 0 to 49907
Data columns (total 19 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   age                              49908 non-null  int64
 1   number_of_dependants             49908 non-null  int64
 2   income_level                     49908 non-null  int64
 3   income_lakhs                     49908 non-null  int64
 4   insurance_plan                   49908 non-null  int64
 5   annual_premium_amount            49908 non-null  int64
 6   total_risk_score                 49908 non-null  int64
 7   gender_Male                      49908 non-null  int64
 8   region_Northwest                 49908 non-null  int64
 9   region_Southeast                 49908 non-null  int64
 10  region_Southwest                 49908 non-null  int64
 11  marital_status_Unmarried         49908 non-null  int64
 12  bmi_category_Obesity             49908 non-null  int64
 13  bmi_category_Overweight          49908 non-null  int64
 14  bmi_category_Underweight         49908 non-null  int64
 15  smoking_status_Occasional        49908 non-null  int64
 16  smoking_status_Regular           49908 non-null  int64
 17  employment_status_Salaried       49908 non-null  int64
 18  employment_status_Self-Employed  49908 non-null  int64
dtypes: int64(19)
memory usage: 7.2 MB
```

[77]:
```
# Showing all the non-encoded columns and one encoded columns from each category

sampled_encoded_cols_index = [0,1,2,3,4,5,6,7,8,11,12,15,17]

df6.iloc[0:5,sampled_encoded_cols_index]
```

[77]:
```
   age  number_of_dependants  income_level  income_lakhs  insurance_plan  \
0   26                     0             1             6               1
1   29                     2             1             6               1
2   49                     2             2            20               2
3   30                     3             4            77               3
4   18                     0             4            99               2

   annual_premium_amount  total_risk_score  gender_Male  region_Northwest  \
0                   9053                 6            1                 1
1                  16339                 6            0                 0
2                  18164                 6            0                 0
3                  20303                 0            0                 0
4                  13365                 6            1                 0

   marital_status_Unmarried  bmi_category_Obesity  smoking_status_Occasional  \
0                         1                     0                          0
1                         0                     1                          0
2                         0                     0                          0
3                         0                     0                          0
4                         1                     0                          0

   employment_status_Salaried
0                           1
1                           1
2                           0
3                           1
4                           0
```

0.4.3 Feature Selection

- To identify the most relevant features, both pairwise correlations and multicollinearity will be analyzed.
- *Correlation* analysis will be used to detect linear relationships, while *Variance Inflation Factor (VIF)* will be employed to identify multicollinearity.
- Prior to these analyses, features will be scaled to ensure comparability across variables.

[78]:
```
# Before Scaling the features

df7 = df6.copy()
df7.sample(3)
```

[78]:
```
       age  number_of_dependants  income_level  income_lakhs  insurance_plan  \
22498   21                     3             3            28               1
6982    20                     1             3            27               1
15687   50                     3             3            26               3

       annual_premium_amount  total_risk_score  gender_Male  region_Northwest  \
22498                   8601                 0            1                 0
6982                    7292                 0            1                 0
15687                  28031                 6            1                 0

       region_Southeast  region_Southwest  marital_status_Unmarried  \
22498                 0                 1                         0
6982                  0                 1                         1
15687                 1                 0                         0

       bmi_category_Obesity  bmi_category_Overweight  \
22498                     0                        0
6982                      0                        0
15687                     0                        0

       bmi_category_Underweight  smoking_status_Occasional  \
22498                         0                          0
6982                          0                          0
15687                         1                          1

       smoking_status_Regular  employment_status_Salaried  \
22498                       0                           0
6982                        1                           0
15687                       0                           1

       employment_status_Self-Employed
22498                                0
6982                                 1
15687                                0
```

[79]:
```
df7.columns
```

[79]:
```
Index(['age', 'number_of_dependants', 'income_level', 'income_lakhs',
       'insurance_plan', 'annual_premium_amount', 'total_risk_score',
       'gender_Male', 'region_Northwest', 'region_Southeast',
       'region_Southwest', 'marital_status_Unmarried', 'bmi_category_Obesity',
       'bmi_category_Overweight', 'bmi_category_Underweight',
       'smoking_status_Occasional', 'smoking_status_Regular',
       'employment_status_Salaried', 'employment_status_Self-Employed'],
      dtype='object')
```

The following columns will be scaled: `age`, `number_of_dependants`, `income_level`, `income_lakhs`, `insurance_plan`, `total_risk_score`.

Scaling type : Minmax scaling

[80]:
```
# Scaling using 'MinMaxScaler'

cols_to_scale = ['age', 'number_of_dependants', 'income_level', 'income_lakhs',␣
↪'insurance_plan', 'total_risk_score']
mms = MinMaxScaler()
df7[cols_to_scale] = mms.fit_transform(df7[cols_to_scale])
```

Correlation

[81]:
```
# Correlation Matrix between features

cr = df7.corr()
cr
```

[81]:
```
                                   age  number_of_dependants  income_level  \
age                           1.000000              0.415742      0.029851
number_of_dependants          0.415742              1.000000      0.006564
income_level                  0.029851              0.006564      1.000000
income_lakhs                  0.025060              0.006074      0.906830
insurance_plan                0.496317              0.256459      0.440428
annual_premium_amount         0.767569              0.414691      0.271811
total_risk_score              0.442773              0.371498      0.013506
gender_Male                  -0.002219             -0.003093      0.063108
region_Northwest              0.000464              0.001693     -0.003324
region_Southeast              0.003305              0.003620     -0.000259
region_Southwest             -0.003424             -0.000339      0.009367
marital_status_Unmarried     -0.543104             -0.841717     -0.012994
bmi_category_Obesity          0.152496              0.115397     -0.002244
bmi_category_Overweight       0.153148              0.110451      0.007947
bmi_category_Underweight     -0.115888             -0.093881      0.000350
smoking_status_Occasional     0.066596              0.071762     -0.001340
smoking_status_Regular        0.059380              0.094829      0.020275
employment_status_Salaried   -0.008093              0.067066     -0.134032
employment_status_Self-Employed  0.314684           0.115930      0.139333

                              income_lakhs  insurance_plan  \
age                               0.025060        0.496317
number_of_dependants              0.006074        0.256459
income_level                      0.906830        0.440428
income_lakhs                      1.000000        0.410753
insurance_plan                    0.410753        1.000000
annual_premium_amount             0.243058        0.834148
total_risk_score                  0.009626        0.260932
gender_Male                       0.039126        0.034211
region_Northwest                 -0.005192       -0.002821
region_Southeast                 -0.001250        0.004082
```

```
region_Southwest                     0.009929        -0.000977
marital_status_Unmarried            -0.011099        -0.316800
bmi_category_Obesity                 0.000314         0.094698
bmi_category_Overweight              0.007150         0.098639
bmi_category_Underweight            -0.000740        -0.073881
smoking_status_Occasional            0.002306         0.037351
smoking_status_Regular               0.010948         0.059587
employment_status_Salaried          -0.100510        -0.041582
employment_status_Self-Employed      0.109759         0.223947

                                 annual_premium_amount  total_risk_score  \
age                                           0.767569          0.442773
number_of_dependants                          0.414691          0.371498
income_level                                  0.271811          0.013506
income_lakhs                                  0.243058          0.009626
insurance_plan                                0.834148          0.260932
annual_premium_amount                         1.000000          0.519458
total_risk_score                              0.519458          1.000000
gender_Male                                   0.064470         -0.003754
region_Northwest                             -0.005078         -0.005627
region_Southeast                              0.008235          0.002019
region_Southwest                             -0.003828         -0.000132
marital_status_Unmarried                     -0.516350         -0.433916
bmi_category_Obesity                          0.249847          0.101039
bmi_category_Overweight                       0.187103          0.102556
bmi_category_Underweight                     -0.135289         -0.087996
smoking_status_Occasional                     0.060610          0.071090
smoking_status_Regular                        0.198829          0.093822
employment_status_Salaried                   -0.005442          0.059511
employment_status_Self-Employed               0.289438          0.135824

                                 gender_Male  region_Northwest  \
age                                -0.002219          0.000464
number_of_dependants               -0.003093          0.001693
income_level                        0.063108         -0.003324
income_lakhs                        0.039126         -0.005192
insurance_plan                      0.034211         -0.002821
annual_premium_amount               0.064470         -0.005078
total_risk_score                   -0.003754         -0.005627
gender_Male                         1.000000         -0.003287
region_Northwest                   -0.003287          1.000000
region_Southeast                   -0.002988         -0.368277
region_Southwest                    0.001873         -0.330654
marital_status_Unmarried            0.003944         -0.001083
bmi_category_Obesity               -0.031351         -0.002262
bmi_category_Overweight             0.080588          0.001194
bmi_category_Underweight           -0.043000         -0.002017
```

smoking_status_Occasional	-0.045618	-0.000669
smoking_status_Regular	0.305180	-0.000255
employment_status_Salaried	0.005559	0.004574
employment_status_Self-Employed	0.001055	0.000110

	region_Southeast	region_Southwest \
age	0.003305	-0.003424
number_of_dependants	0.003620	-0.000339
income_level	-0.000259	0.009367
income_lakhs	-0.001250	0.009929
insurance_plan	0.004082	-0.000977
annual_premium_amount	0.008235	-0.003828
total_risk_score	0.002019	-0.000132
gender_Male	-0.002988	0.001873
region_Northwest	-0.368277	-0.330654
region_Southeast	1.000000	-0.484271
region_Southwest	-0.484271	1.000000
marital_status_Unmarried	-0.003980	0.004287
bmi_category_Obesity	0.006373	-0.000934
bmi_category_Overweight	0.002511	-0.002208
bmi_category_Underweight	0.000211	-0.000499
smoking_status_Occasional	-0.002191	0.006282
smoking_status_Regular	-0.000401	-0.001858
employment_status_Salaried	-0.006575	0.000249
employment_status_Self-Employed	0.003287	-0.005618

	marital_status_Unmarried \
age	-0.543104
number_of_dependants	-0.841717
income_level	-0.012994
income_lakhs	-0.011099
insurance_plan	-0.316800
annual_premium_amount	-0.516350
total_risk_score	-0.433916
gender_Male	0.003944
region_Northwest	-0.001083
region_Southeast	-0.003980
region_Southwest	0.004287
marital_status_Unmarried	1.000000
bmi_category_Obesity	-0.118092
bmi_category_Overweight	-0.117312
bmi_category_Underweight	0.089315
smoking_status_Occasional	-0.075253
smoking_status_Regular	-0.092448
employment_status_Salaried	-0.055285
employment_status_Self-Employed	-0.171646

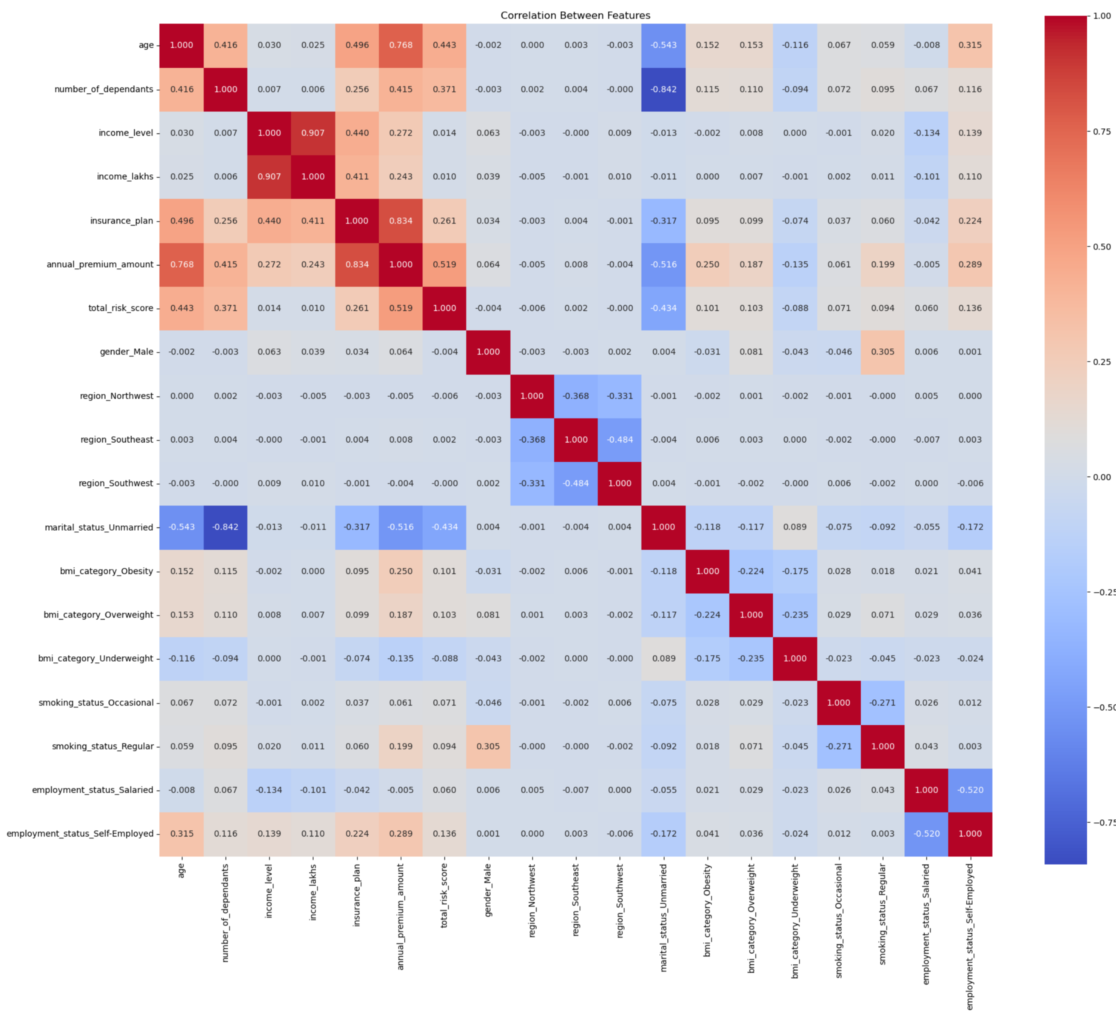
```
                                 bmi_category_Obesity  \
age                                          0.152496
number_of_dependants                         0.115397
income_level                                -0.002244
income_lakhs                                 0.000314
insurance_plan                               0.094698
annual_premium_amount                        0.249847
total_risk_score                             0.101039
gender_Male                                 -0.031351
region_Northwest                            -0.002262
region_Southeast                             0.006373
region_Southwest                            -0.000934
marital_status_Unmarried                    -0.118092
bmi_category_Obesity                         1.000000
bmi_category_Overweight                     -0.224205
bmi_category_Underweight                    -0.175299
smoking_status_Occasional                    0.028321
smoking_status_Regular                       0.017526
employment_status_Salaried                   0.021085
employment_status_Self-Employed              0.040750

                                 bmi_category_Overweight  \
age                                             0.153148
number_of_dependants                            0.110451
income_level                                    0.007947
income_lakhs                                    0.007150
insurance_plan                                  0.098639
annual_premium_amount                           0.187103
total_risk_score                                0.102556
gender_Male                                     0.080588
region_Northwest                                0.001194
region_Southeast                                0.002511
region_Southwest                               -0.002208
marital_status_Unmarried                       -0.117312
bmi_category_Obesity                           -0.224205
bmi_category_Overweight                         1.000000
bmi_category_Underweight                       -0.235191
smoking_status_Occasional                       0.029486
smoking_status_Regular                          0.070944
employment_status_Salaried                      0.028753
employment_status_Self-Employed                 0.035602

                                 bmi_category_Underweight  \
age                                             -0.115888
number_of_dependants                            -0.093881
income_level                                     0.000350
income_lakhs                                    -0.000740
```

```
insurance_plan                                          -0.073881
annual_premium_amount                                   -0.135289
total_risk_score                                        -0.087996
gender_Male                                             -0.043000
region_Northwest                                        -0.002017
region_Southeast                                         0.000211
region_Southwest                                        -0.000499
marital_status_Unmarried                                 0.089315
bmi_category_Obesity                                    -0.175299
bmi_category_Overweight                                 -0.235191
bmi_category_Underweight                                 1.000000
smoking_status_Occasional                               -0.023061
smoking_status_Regular                                  -0.044704
employment_status_Salaried                              -0.023124
employment_status_Self-Employed                         -0.023601

                                 smoking_status_Occasional  \
age                                               0.066596
number_of_dependants                              0.071762
income_level                                     -0.001340
income_lakhs                                      0.002306
insurance_plan                                    0.037351
annual_premium_amount                             0.060610
total_risk_score                                  0.071090
gender_Male                                      -0.045618
region_Northwest                                 -0.000669
region_Southeast                                 -0.002191
region_Southwest                                  0.006282
marital_status_Unmarried                         -0.075253
bmi_category_Obesity                              0.028321
bmi_category_Overweight                           0.029486
bmi_category_Underweight                         -0.023061
smoking_status_Occasional                         1.000000
smoking_status_Regular                           -0.270923
employment_status_Salaried                        0.026424
employment_status_Self-Employed                   0.012346

                                 smoking_status_Regular  \
age                                            0.059380
number_of_dependants                           0.094829
income_level                                   0.020275
income_lakhs                                   0.010948
insurance_plan                                 0.059587
annual_premium_amount                          0.198829
total_risk_score                               0.093822
gender_Male                                    0.305180
region_Northwest                              -0.000255
```

```
region_Southeast                                   -0.000401
region_Southwest                                   -0.001858
marital_status_Unmarried                           -0.092448
bmi_category_Obesity                                0.017526
bmi_category_Overweight                             0.070944
bmi_category_Underweight                           -0.044704
smoking_status_Occasional                          -0.270923
smoking_status_Regular                              1.000000
employment_status_Salaried                          0.042991
employment_status_Self-Employed                     0.002693

                                 employment_status_Salaried  \
age                                               -0.008093
number_of_dependants                               0.067066
income_level                                      -0.134032
income_lakhs                                      -0.100510
insurance_plan                                    -0.041582
annual_premium_amount                             -0.005442
total_risk_score                                   0.059511
gender_Male                                        0.005559
region_Northwest                                   0.004574
region_Southeast                                  -0.006575
region_Southwest                                   0.000249
marital_status_Unmarried                          -0.055285
bmi_category_Obesity                               0.021085
bmi_category_Overweight                            0.028753
bmi_category_Underweight                          -0.023124
smoking_status_Occasional                          0.026424
smoking_status_Regular                             0.042991
employment_status_Salaried                         1.000000
employment_status_Self-Employed                   -0.519576

                                 employment_status_Self-Employed
age                                                     0.314684
number_of_dependants                                    0.115930
income_level                                            0.139333
income_lakhs                                            0.109759
insurance_plan                                          0.223947
annual_premium_amount                                   0.289438
total_risk_score                                        0.135824
gender_Male                                             0.001055
region_Northwest                                        0.000110
region_Southeast                                        0.003287
region_Southwest                                       -0.005618
marital_status_Unmarried                               -0.171646
bmi_category_Obesity                                    0.040750
bmi_category_Overweight                                 0.035602
```

```
bmi_category_Underweight                                    -0.023601
smoking_status_Occasional                                    0.012346
smoking_status_Regular                                       0.002693
employment_status_Salaried                                  -0.519576
employment_status_Self-Employed                              1.000000
```

[82]:
```python
# Correlation Matrix displayed as a Heatmap

plt.figure(figsize=(20,20))
sns.heatmap(cr, annot=True, fmt='.3f', cmap='coolwarm', square=True,␣
 ↪cbar_kws={"shrink": 0.75})
plt.title('Correlation Between Features')
plt.tight_layout()
plt.show()
```



[83]:
```python
# Display features that have a high correlation (|correlation| > 0.35) with
↪'annual_premium_amount'

cr[abs(cr['annual_premium_amount']) > 0.35]['annual_premium_amount']
```

[83]:
```
age                          0.767569
number_of_dependants         0.414691
insurance_plan               0.834148
annual_premium_amount        1.000000
total_risk_score             0.519458
marital_status_Unmarried    -0.516350
Name: annual_premium_amount, dtype: float64
```

[84]:
```python
# Extract the feature names (index labels) that have a high correlation
↪(|correlation| > 0.35) with 'annual_premium_amount'

cr[abs(cr['annual_premium_amount']) > 0.35]['annual_premium_amount'].index
```

[84]:
```
Index(['age', 'number_of_dependants', 'insurance_plan',
       'annual_premium_amount', 'total_risk_score',
       'marital_status_Unmarried'],
      dtype='object')
```

Observation

Based on the correlation heatmap, a strong positive correlation with `annual_premium_amount` is observed for the features `age`, `number_of_dependants`, `insurance_plan`, and `total_risk_score`, suggesting that these variables may significantly influence premium prediction.

Additionally, a strong negative correlation is shown by `marital_status_Unmarried`, indicating that being unmarried is generally associated with lower premium amounts.

Variance Inflation Factor (VIF) The Variance Inflation Factor (VIF) will be checked to detect and address multicollinearity among the features.

[85]:
```python
df7.columns
```

[85]:
```
Index(['age', 'number_of_dependants', 'income_level', 'income_lakhs',
       'insurance_plan', 'annual_premium_amount', 'total_risk_score',
       'gender_Male', 'region_Northwest', 'region_Southeast',
       'region_Southwest', 'marital_status_Unmarried', 'bmi_category_Obesity',
       'bmi_category_Overweight', 'bmi_category_Underweight',
       'smoking_status_Occasional', 'smoking_status_Regular',
       'employment_status_Salaried', 'employment_status_Self-Employed'],
      dtype='object')
```

[86]:
```python
# Calculate VIF for each feature and store the results in a new DataFrame
```

```python
# Initialize a dictionary to store feature names and their corresponding VIF␣
↪scores
vif_dict = {'features':[],'vif_score':[]}

# Exclude the target variable from VIF calculation
temp_df = df7.drop('annual_premium_amount',axis=1)

# Loop through each feature to compute VIF
for i,col in enumerate(temp_df.columns):
    # Calculate the Variance Inflation Factor for the current feature
    vif = variance_inflation_factor(temp_df,i)

    # Append the feature name and its VIF score to the dictionary
    vif_dict['features'].append(col)
    vif_dict['vif_score'].append(vif)
```

[87]:
```python
# Convert the VIF dictionary into a DataFrame for better readability and␣
↪analysis
vif_df = pd.DataFrame(vif_dict)

# Sort features by their VIF scores in descending order to identify highly␣
↪collinear features
vif_df.sort_values(by='vif_score',ascending=False)
```

[87]:

	features	vif_score
2	income_level	12.450675
3	income_lakhs	11.183367
0	age	4.567634
1	number_of_dependants	4.534650
4	insurance_plan	3.584752
10	marital_status_Unmarried	3.411185
8	region_Southeast	2.922414
5	total_risk_score	2.687610
9	region_Southwest	2.670666
6	gender_Male	2.421496
16	employment_status_Salaried	2.382134
17	employment_status_Self-Employed	2.137753
7	region_Northwest	2.102556
15	smoking_status_Regular	1.777089
12	bmi_category_Overweight	1.549922
11	bmi_category_Obesity	1.352806
13	bmi_category_Underweight	1.302886
14	smoking_status_Occasional	1.272745

After the Variance Inflation Factor (VIF) was calculated for all features, it was observed that both `income_level` and `income_lakhs` had VIF scores exceeding the commonly accepted threshold of 10, indicating the presence of high multicollinearity between them.

To address this issue and enhance model stability, the `income_level` feature was dropped, as it exhibited the higher VIF value of the two. This step was taken to reduce redundancy without significantly compromising the information contained in the dataset.

[88]:
```python
# Initialize a list to store highly collinear features to be removed later

high_vif_features  = []
```

[89]:
```python
# Identify the feature with the highest VIF score and add it to the removal␣
↪list 'high_vif_features'

highest_vif_feature = vif_df['features'][vif_df['vif_score'].idxmax()]
high_vif_features.append(highest_vif_feature)
print(high_vif_features)
```

```
['income_level']
```

[90]:
```python
# Drop the most collinear feature from the temporary DataFrame

temp_df1 = temp_df.drop(['income_level'],axis=1).copy()
temp_df1
```

[90]:
```
            age  number_of_dependants  income_lakhs  insurance_plan  \
0      0.148148                   0.0      0.050505             0.0
1      0.203704                   0.4      0.050505             0.0
2      0.574074                   0.4      0.191919             0.5
3      0.222222                   0.6      0.767677             1.0
4      0.000000                   0.0      0.989899             0.5
…           …                    …           …            …
49903  0.111111                   0.0      0.343434             0.0
49904  0.537037                   0.4      0.818182             1.0
49905  0.055556                   0.0      0.313131             0.0
49906  0.000000                   0.4      0.191919             0.0
49907  0.555556                   0.6      0.060606             0.5

       total_risk_score  gender_Male  region_Northwest  region_Southeast  \
0              0.428571            1                 1                 0
1              0.428571            0                 0                 1
2              0.428571            0                 0                 0
3              0.000000            0                 0                 1
4              0.428571            1                 0                 0
…                   …          …                …                 …
49903          0.000000            0                 1                 0
49904          0.357143            0                 0                 1
49905          0.000000            1                 1                 0
49906          0.000000            1                 1                 0
49907          0.428571            0                 0                 0
```

```
       region_Southwest  marital_status_Unmarried  bmi_category_Obesity  \
0                     0                         1                     0
1                     0                         0                     1
2                     0                         0                     0
3                     0                         0                     0
4                     0                         1                     0
…                   …                         …                   …
49903                 0                         1                     0
49904                 0                         0                     0
49905                 0                         1                     0
49906                 0                         1                     0
49907                 1                         0                     0

       bmi_category_Overweight  bmi_category_Underweight  \
0                            0                         0
1                            0                         0
2                            0                         0
3                            0                         0
4                            1                         0
…                          …                         …
49903                        0                         1
49904                        0                         0
49905                        0                         0
49906                        0                         0
49907                        0                         0

       smoking_status_Occasional  smoking_status_Regular  \
0                              0                       0
1                              0                       1
2                              0                       0
3                              0                       0
4                              0                       1
…                            …                       …
49903                          0                       0
49904                          0                       0
49905                          0                       1
49906                          0                       0
49907                          1                       0

       employment_status_Salaried  employment_status_Self-Employed
0                               1                                0
1                               1                                0
2                               0                                1
3                               1                                0
4                               0                                1
…                             …                              …
49903                           0                                1
```

```
49904                               1                                0
49905                               0                                0
49906                               1                                0
49907                               0                                1

[49908 rows x 17 columns]
```

The same process will be repeated until all VIF values are within the acceptable threshold.

[91]:
```python
# Recalculate VIF scores for the updated feature set

vif_dict = {'features':[],'vif_score':[]}

for i,col in enumerate(temp_df1.columns):
    vif = variance_inflation_factor(temp_df1,i)
    vif_dict['features'].append(col)
    vif_dict['vif_score'].append(vif)

# Create a DataFrame from the updated VIF scores
vif_df = pd.DataFrame(vif_dict)

# Sort VIF scores in descending order to identify next candidates for removal
vif_df.sort_values(by='vif_score',ascending=False)
```

[91]:

	features	vif_score
0	age	4.545825
1	number_of_dependants	4.526598
3	insurance_plan	3.445682
9	marital_status_Unmarried	3.393718
7	region_Southeast	2.919775
4	total_risk_score	2.687326
8	region_Southwest	2.668314
2	income_lakhs	2.480563
5	gender_Male	2.409980
15	employment_status_Salaried	2.374628
16	employment_status_Self-Employed	2.132810
6	region_Northwest	2.100789
14	smoking_status_Regular	1.777024
11	bmi_category_Overweight	1.549907
10	bmi_category_Obesity	1.352748
12	bmi_category_Underweight	1.302636
13	smoking_status_Occasional	1.272744

All remaining features have acceptable VIF scores.

[92]:
```python
# Remove the previously identified high-VIF features from the original␣
↪DataFrame.
final_df = df7.drop(high_vif_features,axis=1)
```

```python
# Display the final cleaned dataset
final_df
```

[92]:

```
            age  number_of_dependants  income_lakhs  insurance_plan  \
0      0.148148                   0.0      0.050505             0.0
1      0.203704                   0.4      0.050505             0.0
2      0.574074                   0.4      0.191919             0.5
3      0.222222                   0.6      0.767677             1.0
4      0.000000                   0.0      0.989899             0.5
…           …                     …             …               …
49903  0.111111                   0.0      0.343434             0.0
49904  0.537037                   0.4      0.818182             1.0
49905  0.055556                   0.0      0.313131             0.0
49906  0.000000                   0.4      0.191919             0.0
49907  0.555556                   0.6      0.060606             0.5

       annual_premium_amount  total_risk_score  gender_Male  region_Northwest  \
0                       9053          0.428571            1                 1
1                      16339          0.428571            0                 0
2                      18164          0.428571            0                 0
3                      20303          0.000000            0                 0
4                      13365          0.428571            1                 0
…                          …                 …            …                 …
49903                   9111          0.000000            0                 1
49904                  27076          0.357143            0                 0
49905                   8564          0.000000            1                 1
49906                   9490          0.000000            1                 1
49907                  19730          0.428571            0                 0

       region_Southeast  region_Southwest  marital_status_Unmarried  \
0                     0                 0                         1
1                     1                 0                         0
2                     0                 0                         0
3                     1                 0                         0
4                     0                 0                         1
…                     …                 …                         …
49903                 0                 0                         1
49904                 1                 0                         0
49905                 0                 0                         1
49906                 0                 0                         1
49907                 0                 1                         0

       bmi_category_Obesity  bmi_category_Overweight  \
0                         0                        0
1                         1                        0
2                         0                        0
```

```
3                          0                         0
4                          0                         1
…                      …                       …
49903                      0                         0
49904                      0                         0
49905                      0                         0
49906                      0                         0
49907                      0                         0

       bmi_category_Underweight  smoking_status_Occasional  \
0                             0                          0
1                             0                          0
2                             0                          0
3                             0                          0
4                             0                          0
…                         …                          …
49903                         1                          0
49904                         0                          0
49905                         0                          0
49906                         0                          0
49907                         0                          1

       smoking_status_Regular  employment_status_Salaried  \
0                           0                           1
1                           1                           1
2                           0                           0
3                           0                           1
4                           1                           0
…                       …                          …
49903                       0                           0
49904                       0                           1
49905                       1                           0
49906                       0                           1
49907                       0                           0

       employment_status_Self-Employed
0                                    0
1                                    0
2                                    1
3                                    0
4                                    1
…                                …
49903                                1
49904                                0
49905                                0
49906                                0
49907                                1
```

```
[49908 rows x 18 columns]
```

0.5 Model Selection

The model selection process proceeded as follows:

1. Split the dataset into training and test sets.
2. Trained different models on the training data.
3. Used cross-validation to compare model performance.
4. Tuned hyperparameters of the best model.
5. Evaluated the final model's performance on the test set.

0.5.1 Dataset Split

The dataset is divided as follows:

- Training set: 70%
- Test set: 30%

Split was done randomly with a fixed seed for reproducibility.

[93]:
```
# Separate features and target variable

features = final_df.drop(['annual_premium_amount'],axis=1)
target = final_df['annual_premium_amount']
```

[94]:
```
# Split the data into training and test sets

X_train,X_test,y_train,y_test = train_test_split(features,target,test_size=0.
↪3,random_state=42)
```

[95]:
```
# Display the shape of training and test sets

print(f"X_train shape: {X_train.shape}, y_train shape: {y_train.shape}")
print(f"X_test shape: {X_test.shape}, y_test shape: {y_test.shape}")
```

```
X_train shape: (34935, 17), y_train shape: (34935,)
X_test shape: (14973, 17), y_test shape: (14973,)
```

0.5.2 Model Training

Various models will be tried and training will be performed, including:

- Linear Regression

- Ridge Regression
- Lasso Regression
- Random Forest Regressor
- XGBoost Regressor

Linear Regression

[96]:
```
# Initialize and train the Linear Regression model
lr=LinearRegression()
lr.fit(X_train,y_train)

# Evaluate model performance
train_score = lr.score(X_train,y_train)
test_score = lr.score(X_test,y_test)

# Print the R² scores
print(f'Train Score : {train_score} , Test Score : {test_score} ')
```

```
Train Score : 0.9280957176093705 , Test Score : 0.9283765993531427
```

[97]:
```
# Predict on test data
y_pred = lr.predict(X_test)

# Calculate Mean Squared Error and Root Mean Squared Error
mse = mean_squared_error(y_test,y_pred)
rmse = root_mean_squared_error(y_test,y_pred)

# Print performance metrics
print(f'MSE : {mse} , RMSE : {rmse}')
```

```
MSE : 5056639.130347778 , RMSE : 2248.697207350909
```

[98]:
```
# Feature Names and Its Coefficents

print('Features -> ',lr.feature_names_in_)
print('\n','**'*50)
print('Coef -> ',lr.coef_)
```

```
Features ->  ['age' 'number_of_dependants' 'income_lakhs' 'insurance_plan'
 'total_risk_score' 'gender_Male' 'region_Northwest' 'region_Southeast'
 'region_Southwest' 'marital_status_Unmarried' 'bmi_category_Obesity'
 'bmi_category_Overweight' 'bmi_category_Underweight'
 'smoking_status_Occasional' 'smoking_status_Regular'
 'employment_status_Salaried' 'employment_status_Self-Employed']

 ********************************************************************************
********************
```

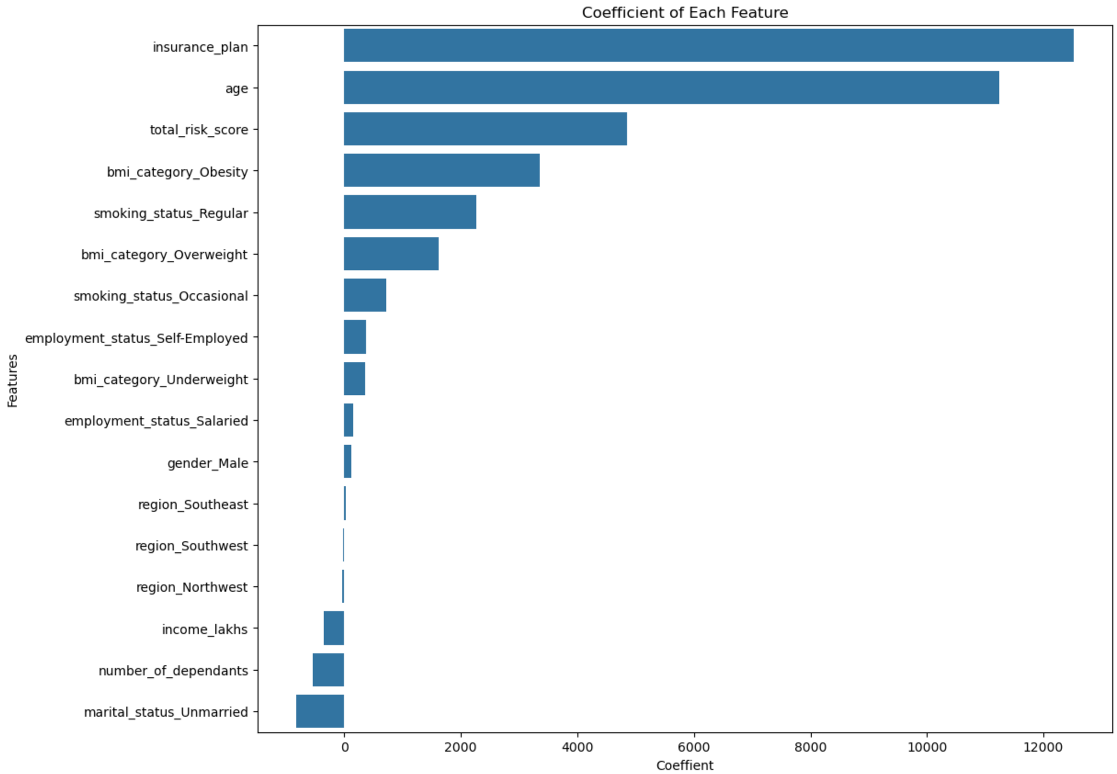
```
Coef ->  [11236.4082633   -536.61172956  -353.97067084 12515.43782972
  4846.88626348   121.53023031   -34.50219135    27.97504587
   -23.47372614  -821.78088646  3356.07750448  1613.1421694
   364.31350886   722.41051531  2261.97816483   149.57179545
   378.25130285]
```

[99]:
```
# Creating a dataframe of features and coefficients

feat_coef = {
    'features' : lr.feature_names_in_,
    'coef' : lr.coef_
}

feat_coef_df = pd.DataFrame(feat_coef)
feat_coef_df.sort_values(by=['coef'],ascending=False)
```

[99]:

	features	coef
3	insurance_plan	12515.437830
0	age	11236.408263
4	total_risk_score	4846.886263
10	bmi_category_Obesity	3356.077504
14	smoking_status_Regular	2261.978165
11	bmi_category_Overweight	1613.142169
13	smoking_status_Occasional	722.410515
16	employment_status_Self-Employed	378.251303
12	bmi_category_Underweight	364.313509
15	employment_status_Salaried	149.571795
5	gender_Male	121.530230
7	region_Southeast	27.975046
8	region_Southwest	-23.473726
6	region_Northwest	-34.502191
2	income_lakhs	-353.970671
1	number_of_dependants	-536.611730
9	marital_status_Unmarried	-821.780886

[100]:
```
# Plotting the features and its coefficients

plt.figure(figsize=(12,10))
sns.barplot(data=feat_coef_df.sort_values(by=['coef'],ascending=False),x =␣
↪'coef',y = 'features' )
plt.title('Coefficient of Each Feature')
plt.xlabel('Coeffient')
plt.ylabel('Features')
plt.show()
```



Lasso Regression

[101]:
```
# Initialize and train the Lasso Regression model
ls = Lasso()
ls.fit(X_train,y_train)

# Evaluate model performance
train_score = ls.score(X_train,y_train)
test_score = ls.score(X_test,y_test)

# Print the R² scores
print(f'Train Score : {train_score} , Test Score : {test_score} ')
```

```
Train Score : 0.9280927085624446 , Test Score : 0.9283637752728616
```

[102]:
```
# Predict on test data
y_pred = ls.predict(X_test)

# Calculate Mean Squared Error and Root Mean Squared Error
mse = mean_squared_error(y_test,y_pred)
rmse = root_mean_squared_error(y_test,y_pred)

# Print performance metrics
```

```
print(f'MSE : {mse} , RMSE : {rmse}')
```

```
MSE : 5057544.515257937 , RMSE : 2248.8985115513633
```

Ridge Regression

[103]:
```
# Initialize and train the Ridge Regression model
rg = Ridge()
rg.fit(X_train,y_train)

# Evaluate model performance
train_score = rg.score(X_train,y_train)
test_score = rg.score(X_test,y_test)

# Print the R² scores
print(f'Train Score : {train_score} , Test Score : {test_score} ')
```

```
Train Score : 0.9280956798900618 , Test Score : 0.9283764905001947
```

[104]:
```
# Predict on test data
y_pred = rg.predict(X_test)

# Calculate Mean Squared Error and Root Mean Squared Error
mse = mean_squared_error(y_test,y_pred)
rmse = root_mean_squared_error(y_test,y_pred)

# Print performance metrics
print(f'MSE : {mse} , RMSE : {rmse}')
```

```
MSE : 5056646.815407011 , RMSE : 2248.698916130617
```

Observation

Neither Lasso nor Ridge regression showed significant improvement over Linear Regression.

Random Forest Regressor

[105]:
```
# Initialize and train the Random Forest Regression model
rfr = RandomForestRegressor()
rfr.fit(X_train,y_train)

# Evaluate model performance
train_score = rfr.score(X_train,y_train)
test_score = rfr.score(X_test,y_test)

# Print the R² scores
print(f'Train Score : {train_score} , Test Score : {test_score} ')
```

```
Train Score : 0.9965657768248808 , Test Score : 0.9789939664693872
```

[106]:
```
# Predict on test data
y_pred = rfr.predict(X_test)

# Calculate Mean Squared Error and Root Mean Squared Error
mse = mean_squared_error(y_test,y_pred)
rmse = root_mean_squared_error(y_test,y_pred)

# Print performance metrics
print(f'MSE : {mse} , RMSE : {rmse}')
```

```
MSE : 1483033.8990467205 , RMSE : 1217.798792513246
```

Based on the R^2 score, it can be seen that this model fits the data well. The XGBoost model will be tried next.

XGboost Regressor

[107]:
```
# Initialize and train the XGboost model
xgb = XGBRegressor()
xgb.fit(X_train,y_train)

# Evaluate model performance
train_score = xgb.score(X_train,y_train)
test_score = xgb.score(X_test,y_test)

# Print the R² scores
print(f'Train Score : {train_score} , Test Score : {test_score} ')
```

```
Train Score : 0.9861041903495789 , Test Score : 0.9807721972465515
```

[108]:
```
# Predict on test data
y_pred = xgb.predict(X_test)

# Calculate Mean Squared Error and Root Mean Squared Error
mse = mean_squared_error(y_test,y_pred)
rmse = root_mean_squared_error(y_test,y_pred)

# Print performance metrics
print(f'MSE : {mse} , RMSE : {rmse}')
```

```
MSE : 1357488.75 , RMSE : 1165.1131591796875
```

Model Performance Comparison: XGBoost vs. Random Forest

Both XGBoost and Random Forest models are observed to perform well on the dataset. However, the following points are noted:

- A lower training score and a higher test score are yielded by the XGBoost model compared to Random Forest.
- This suggests that the XGBoost model generalizes better and is less prone to overfitting.
- To validate this observation, cross-validation will be used to evaluate both models more robustly using `cross_validate`.

Cross Validation - RF vs XGboost Stratified K-Fold will not be performed since this is a regression task; therefore, K-Fold cross-validation is considered sufficient.

[109]:
```
# Set up 5-fold cross-validation

kf = KFold(n_splits = 5,shuffle=True,random_state=42)
```

Cross-validation will be performed on both Random Forest and XGBoost models, with the run time also being recorded.
The `cross_validate` function is used instead of `cross_val_score` to obtain both training and testing scores.

[110]:
```
# Perform cross-validation for Random Forest Regressor

start_time = time()
cv_rfr =␣
 ↪cross_validate(rfr,features,target,cv=kf,return_train_score=True,scoring='r2')
end_time = time()
total_time_rfr = end_time - start_time
print(f'Total Time Taken : {round(total_time_rfr,2)} seconds')
```

```
Total Time Taken : 67.49 seconds
```

[111]:
```
# Display training scores
print("Training scores (R²):", cv_rfr['train_score'])

# Display test scores
print("Validation scores (R²):", cv_rfr['test_score'])
```

```
Training scores (R²): [0.9965103  0.99652913 0.99644755 0.99654474 0.99645785]
Validation scores (R²): [0.97901928 0.97902788 0.97945362 0.97865826 0.9794153 ]
```

[112]:
```
# Perform cross-validation for XGboost Regressor

start_time = time()
cv_xgb =␣
 ↪cross_validate(xgb,features,target,cv=kf,return_train_score=True,scoring='r2')
end_time = time()
total_time_xgb = end_time - start_time
print(f'Total Time Taken : {round(total_time_xgb,2)} seconds')
```

```
Total Time Taken : 0.74 seconds
```

[113]:
```
# Display training scores
print("Training scores (R²):", cv_xgb['train_score'])

# Display test scores
print("Validation scores (R²):", cv_xgb['test_score'])
```

```
Training scores (R²): [0.98580199 0.9857831  0.98563534 0.98585421 0.98580426]
Validation scores (R²): [0.9809615  0.98077649 0.98098087 0.98112887 0.98120141]
```

[114]:
```python
cv_df = pd.DataFrame(
    {
        'model' : ['Random Forest','XGboost'],
        'execution_time (seconds)' : [total_time_rfr,total_time_xgb],
        'average_train_score' : [np.mean(cv_rfr['train_score']),np.
 ↪mean(cv_xgb['train_score'])],
        'average_test_score' : [np.mean(cv_rfr['test_score']),np.
 ↪mean(cv_xgb['test_score'])]
    }
)

cv_df
```

[114]:
```
           model  execution_time (seconds)  average_train_score  \
0  Random Forest                 67.486237             0.996498
1        XGboost                  0.744500             0.985776

   average_test_score
0            0.979115
1            0.981010
```

Cross-Validation Summary: XGBoost vs. Random Forest

The cross-validation results confirm that XGBoost is consistently observed to generalize better than the Random Forest model:

- *Lower training scores* and *higher test scores* were achieved by XGBoost, indicating *better generalization* and *reduced overfitting*.
- In terms of performance, XGBoost was approximately ***83 times faster*** than Random Forest during cross-validation.

Given its superior predictive performance and computational efficiency, **XGBoost will be selected as the final model**.