LUNG-CANCER-PRED-ML-&-CAUSUAL_iNFERENCE

April 3, 2025

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/lung-disease-data/Lung Cancer Dataset.csv
```

!pip install pgmpy

[3]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import tensorflow as tf
import statsmodels.api as sm


from pandas.plotting import scatter_matrix
from scipy.stats import skew , zscore
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from tensorflow import keras
from sklearn.impute import SimpleImputer
from statsmodels.stats.outliers_influence import variance_inflation_factor
from scipy.stats import shapiro, kstest, anderson
from statsmodels.stats.stattools import durbin_watson
from statsmodels.stats.outliers_influence import OLSInfluence
from sklearn.metrics import  precision_score, recall_score, f1_score
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

[4]:
```
#Read the csv dataset file:
data = pd.read_csv("/kaggle/input/lung-disease-data/Lung Cancer Dataset.csv")
```

[5]:
```
# View First Few Data:
data.head()
```

[5]:

	AGE	GENDER	SMOKING	FINGER_DISCOLORATION	MENTAL_STRESS	EXPOSURE_TO_POLLUTION	LONG_TERM_ILLNESS	ENERGY_LEVEL	IMMUNE_WEAKNESS	BREATHING_ISSUE	ALCOHOL_CONSUMPTION	THROAT_DISCOMFORT	OXYGEN_SATURATION
0	68	1	1	1	1	1	0	57.831178	0	0	1	1	95.977287
1	81	1	1	0	0	1	1	47.694835	1	1	0	1	97.184483
2	58	1	1	0	0	0	0	59.577435	0	1	1	0	94.974939
3	44	0	1	0	1	1	0	59.785767	0	1	0	1	95.187900
4	72	0	1	1	1	1	1	59.733941	0	1	0	1	93.503008

```
   CHEST_TIGHTNESS  FAMILY_HISTORY  SMOKING_FAMILY_HISTORY  STRESS_IMMUNE  \
0                1               0                       0              0
1                0               0                       0              0
2                0               0                       0              0
3                0               0                       0              0
4                0               0                       0              0

  PULMONARY_DISEASE
0                NO
1               YES
2                NO
3               YES
4               YES
```

[6]:
```
#View Last Few Data:
data.tail()
```

[6]:
```
      AGE  GENDER  SMOKING  FINGER_DISCOLORATION  MENTAL_STRESS  \
4995   32       0        1                     1              0
4996   80       0        1                     1              1
4997   51       1        0                     0              1
4998   76       1        0                     1              0
4999   33       0        1                     0              0

      EXPOSURE_TO_POLLUTION  LONG_TERM_ILLNESS  ENERGY_LEVEL  IMMUNE_WEAKNESS  \
4995                      0                  1     60.700696                1
4996                      1                  1     50.751741                0
4997                      0                  0     61.063496                1
4998                      0                  0     48.662872                0
4999                      1                  1     58.245188                0

      BREATHING_ISSUE  ALCOHOL_CONSUMPTION  THROAT_DISCOMFORT  \
4995                1                    1                  1
4996                1                    1                  1
4997                0                    0                  0
4998                1                    0                  1
4999                1                    1                  1

      OXYGEN_SATURATION  CHEST_TIGHTNESS  FAMILY_HISTORY  \
4995          94.012495                0               1
4996          94.394968                0               0
4997          98.108901                0               0
4998          95.577773                1               0
4999          94.206934                1               0

      SMOKING_FAMILY_HISTORY  STRESS_IMMUNE PULMONARY_DISEASE
```

```
4995                        1                0              YES
4996                        0                0              YES
4997                        0                1               NO
4998                        0                0               NO
4999                        0                0               NO
```

[7]:
```
# Summary of the Dataset:
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   AGE                     5000 non-null   int64  
 1   GENDER                  5000 non-null   int64  
 2   SMOKING                 5000 non-null   int64  
 3   FINGER_DISCOLORATION    5000 non-null   int64  
 4   MENTAL_STRESS           5000 non-null   int64  
 5   EXPOSURE_TO_POLLUTION   5000 non-null   int64  
 6   LONG_TERM_ILLNESS       5000 non-null   int64  
 7   ENERGY_LEVEL            5000 non-null   float64
 8   IMMUNE_WEAKNESS         5000 non-null   int64  
 9   BREATHING_ISSUE         5000 non-null   int64  
 10  ALCOHOL_CONSUMPTION     5000 non-null   int64  
 11  THROAT_DISCOMFORT       5000 non-null   int64  
 12  OXYGEN_SATURATION       5000 non-null   float64
 13  CHEST_TIGHTNESS         5000 non-null   int64  
 14  FAMILY_HISTORY          5000 non-null   int64  
 15  SMOKING_FAMILY_HISTORY  5000 non-null   int64  
 16  STRESS_IMMUNE           5000 non-null   int64  
 17  PULMONARY_DISEASE       5000 non-null   object 
dtypes: float64(2), int64(15), object(1)
memory usage: 703.2+ KB
```

[8]:
```
print(data['PULMONARY_DISEASE'].unique())
```

```
['NO' 'YES']
```

[9]:
```
# Pulmonary Disease Yes NO to 1 0 integer conversion to uniformize datatype:
# Convert to numeric binary :
# Map values
data['PULMONARY_DISEASE'] = data['PULMONARY_DISEASE'].map({'YES': 1, 'NO': 0})

# Verify
print(data['PULMONARY_DISEASE'].value_counts())
```

```
PULMONARY_DISEASE
0    2963
```

```
1    2037
Name: count, dtype: int64
```

[10]:
```
# Summary Statistics :
data.describe()
```

[10]:

	AGE	GENDER	SMOKING	FINGER_DISCOLORATION	\
count	5000.000000	5000.000000	5000.000000	5000.0000	
mean	57.222800	0.501200	0.666400	0.6012	
std	15.799224	0.500049	0.471546	0.4897	
min	30.000000	0.000000	0.000000	0.0000	
25%	44.000000	0.000000	0.000000	0.0000	
50%	57.000000	1.000000	1.000000	1.0000	
75%	71.000000	1.000000	1.000000	1.0000	
max	84.000000	1.000000	1.000000	1.0000	

	MENTAL_STRESS	EXPOSURE_TO_POLLUTION	LONG_TERM_ILLNESS	ENERGY_LEVEL	\
count	5000.000000	5000.000000	5000.000000	5000.000000	
mean	0.539800	0.516000	0.439200	55.032043	
std	0.498463	0.499794	0.496339	7.913083	
min	0.000000	0.000000	0.000000	23.258308	
25%	0.000000	0.000000	0.000000	49.440685	
50%	1.000000	1.000000	0.000000	55.050421	
75%	1.000000	1.000000	1.000000	60.323320	
max	1.000000	1.000000	1.000000	83.046971	

	IMMUNE_WEAKNESS	BREATHING_ISSUE	ALCOHOL_CONSUMPTION	\
count	5000.000000	5000.00000	5000.000000	
mean	0.394800	0.80040	0.354200	
std	0.488857	0.39974	0.478318	
min	0.000000	0.00000	0.000000	
25%	0.000000	1.00000	0.000000	
50%	0.000000	1.00000	0.000000	
75%	1.000000	1.00000	1.000000	
max	1.000000	1.00000	1.000000	

	THROAT_DISCOMFORT	OXYGEN_SATURATION	CHEST_TIGHTNESS	FAMILY_HISTORY	\
count	5000.000000	5000.000000	5000.000000	5000.000000	
mean	0.698200	94.991029	0.600600	0.301800	
std	0.459085	1.481048	0.489824	0.459085	
min	0.000000	89.923133	0.000000	0.000000	
25%	0.000000	93.973176	0.000000	0.000000	
50%	1.000000	94.974073	1.000000	0.000000	
75%	1.000000	95.989272	1.000000	1.000000	
max	1.000000	99.795786	1.000000	1.000000	

	SMOKING_FAMILY_HISTORY	STRESS_IMMUNE	PULMONARY_DISEASE

```
count             5000.000000    5000.000000          5000.0000
mean                 0.204000       0.209600             0.4074
std                  0.403009       0.407064             0.4914
min                  0.000000       0.000000             0.0000
25%                  0.000000       0.000000             0.0000
50%                  0.000000       0.000000             0.0000
75%                  0.000000       0.000000             1.0000
max                  1.000000       1.000000             1.0000
```

[11]:
```
#Check data type if converted to same (Yes)
print(data.dtypes)
```

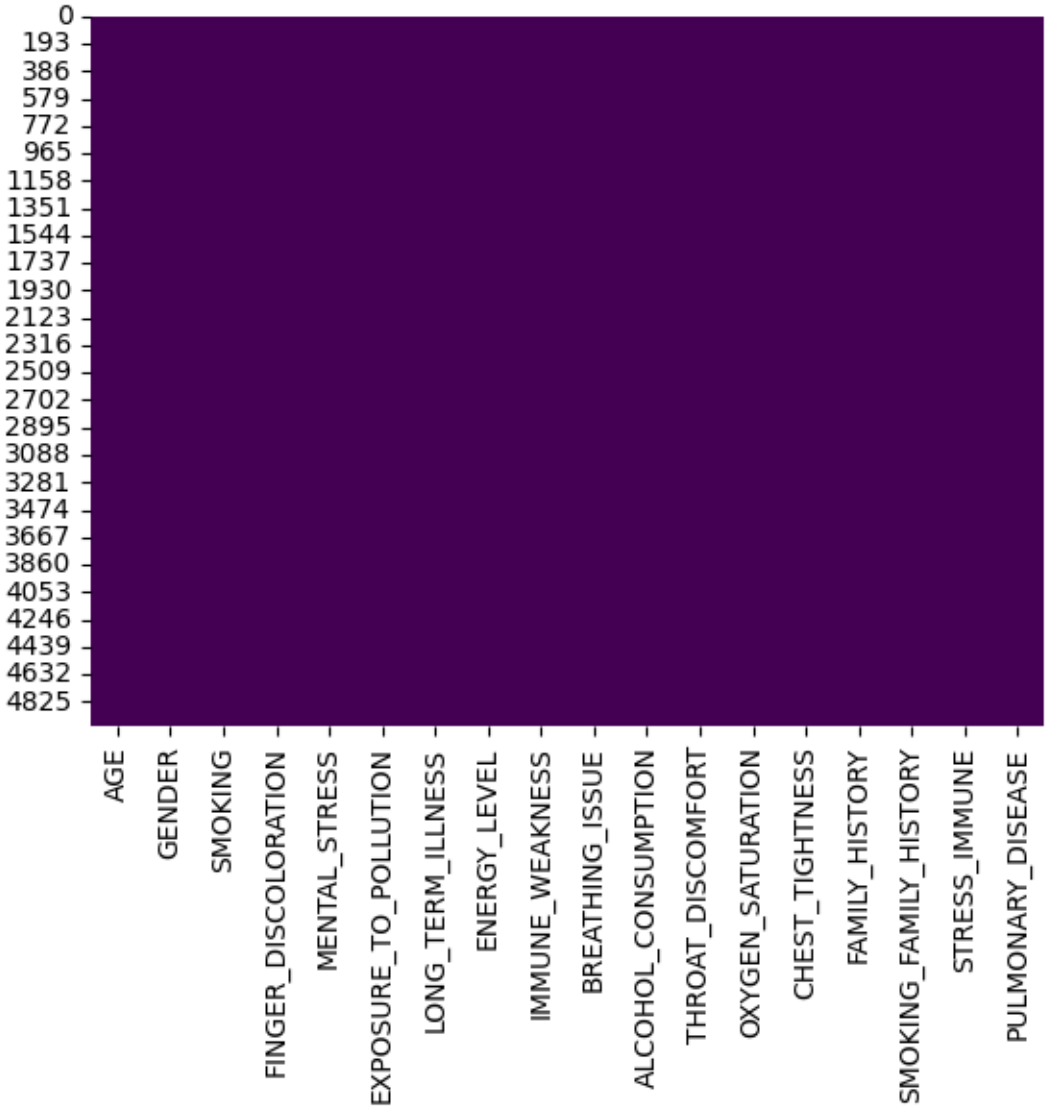
```
AGE                          int64
GENDER                       int64
SMOKING                      int64
FINGER_DISCOLORATION         int64
MENTAL_STRESS                int64
EXPOSURE_TO_POLLUTION        int64
LONG_TERM_ILLNESS            int64
ENERGY_LEVEL               float64
IMMUNE_WEAKNESS              int64
BREATHING_ISSUE              int64
ALCOHOL_CONSUMPTION          int64
THROAT_DISCOMFORT            int64
OXYGEN_SATURATION          float64
CHEST_TIGHTNESS              int64
FAMILY_HISTORY               int64
SMOKING_FAMILY_HISTORY       int64
STRESS_IMMUNE                int64
PULMONARY_DISEASE            int64
dtype: object
```

[12]:
```
# Check for null values :
data.isnull().sum()
```

[12]:
```
AGE                          0
GENDER                       0
SMOKING                      0
FINGER_DISCOLORATION         0
MENTAL_STRESS                0
EXPOSURE_TO_POLLUTION        0
LONG_TERM_ILLNESS            0
ENERGY_LEVEL                 0
IMMUNE_WEAKNESS              0
BREATHING_ISSUE              0
ALCOHOL_CONSUMPTION          0
THROAT_DISCOMFORT            0
OXYGEN_SATURATION            0
```

```
CHEST_TIGHTNESS           0
FAMILY_HISTORY            0
SMOKING_FAMILY_HISTORY    0
STRESS_IMMUNE             0
PULMONARY_DISEASE         0
dtype: int64
```

[13]:
```
sns.heatmap(data.isnull(), cbar=False, cmap='viridis')
plt.show()
```



[14]:
```
# No null value , incase we have null value we need to handle them by␣
↪appropiate imputation techniqes.
```

```python
# Or sometimes dropping certain unnecessary rows.
# Now check for Duplicates
data.duplicated()
data.duplicated().sum()
# Check if any row is duplicated (returns True/False)
has_duplicates = data.duplicated().any()
print("Has duplicates:", has_duplicates)  # Output: False (if all columns are␣
 ↪checked)
duplicate_rows = data[data.duplicated(keep=False)]  # `keep=False` marks all␣
 ↪duplicates
print("All duplicate rows:\n", duplicate_rows)
# Remove all rows that have duplicates (keep only unique rows)
data_cleaned = data.drop_duplicates(keep=False)
print("DataFrame with no duplicates at all:\n", data_cleaned)
```

```
Has duplicates: False
All duplicate rows:
 Empty DataFrame
Columns: [AGE, GENDER, SMOKING, FINGER_DISCOLORATION, MENTAL_STRESS,
EXPOSURE_TO_POLLUTION, LONG_TERM_ILLNESS, ENERGY_LEVEL, IMMUNE_WEAKNESS,
BREATHING_ISSUE, ALCOHOL_CONSUMPTION, THROAT_DISCOMFORT, OXYGEN_SATURATION,
CHEST_TIGHTNESS, FAMILY_HISTORY, SMOKING_FAMILY_HISTORY, STRESS_IMMUNE,
PULMONARY_DISEASE]
Index: []
DataFrame with no duplicates at all:
       AGE  GENDER  SMOKING  FINGER_DISCOLORATION  MENTAL_STRESS  \
0      68       1        1                     1              1
1      81       1        1                     0              0
2      58       1        1                     0              0
3      44       0        1                     0              1
4      72       0        1                     1              1
…   …     …       …                     …              …
4995   32       0        1                     1              0
4996   80       0        1                     1              1
4997   51       1        0                     0              1
4998   76       1        0                     1              0
4999   33       0        1                     0              0

      EXPOSURE_TO_POLLUTION  LONG_TERM_ILLNESS  ENERGY_LEVEL  IMMUNE_WEAKNESS  \
0                         1                  0     57.831178                0
1                         1                  1     47.694835                1
2                         0                  0     59.577435                0
3                         1                  0     59.785767                0
4                         1                  1     59.733941                0
…                       …                  …           …                …
4995                      0                  1     60.700696                1
4996                      1                  1     50.751741                0
```

```
4997                    0                 0     61.063496                1
4998                    0                 0     48.662872                0
4999                    1                 1     58.245188                0

      BREATHING_ISSUE  ALCOHOL_CONSUMPTION  THROAT_DISCOMFORT  \
0                   0                    1                  1
1                   1                    0                  1
2                   1                    1                  0
3                   1                    0                  1
4                   1                    0                  1
…                 …                    …                  …
4995                1                    1                  1
4996                1                    1                  1
4997                0                    0                  0
4998                1                    0                  1
4999                1                    1                  1

      OXYGEN_SATURATION  CHEST_TIGHTNESS  FAMILY_HISTORY  \
0             95.977287                1               0
1             97.184483                0               0
2             94.974939                0               0
3             95.187900                0               0
4             93.503008                0               0
…                   …                …               …
4995          94.012495                0               1
4996          94.394968                0               0
4997          98.108901                0               0
4998          95.577773                1               0
4999          94.206934                1               0

      SMOKING_FAMILY_HISTORY  STRESS_IMMUNE  PULMONARY_DISEASE
0                          0              0                  0
1                          0              0                  1
2                          0              0                  0
3                          0              0                  1
4                          0              0                  1
…                        …              …                  …
4995                       1              0                  1
4996                       0              0                  1
4997                       0              1                  0
4998                       0              0                  0
4999                       0              0                  0

[5000 rows x 18 columns]
```

[15]:
```
# Columns of our dataset:
data_cleaned.columns
```

[15]:
```
Index(['AGE', 'GENDER', 'SMOKING', 'FINGER_DISCOLORATION', 'MENTAL_STRESS',
       'EXPOSURE_TO_POLLUTION', 'LONG_TERM_ILLNESS', 'ENERGY_LEVEL',
       'IMMUNE_WEAKNESS', 'BREATHING_ISSUE', 'ALCOHOL_CONSUMPTION',
       'THROAT_DISCOMFORT', 'OXYGEN_SATURATION', 'CHEST_TIGHTNESS',
       'FAMILY_HISTORY', 'SMOKING_FAMILY_HISTORY', 'STRESS_IMMUNE',
       'PULMONARY_DISEASE'],
      dtype='object')
```

[16]:
```
Indep=data_cleaned.drop(columns=["PULMONARY_DISEASE"])
Indep
```

[16]:

	AGE	GENDER	SMOKING	FINGER_DISCOLORATION	MENTAL_STRESS	\
0	68	1	1	1	1	
1	81	1	1	0	0	
2	58	1	1	0	0	
3	44	0	1	0	1	
4	72	0	1	1	1	
…	…	…	…	…	…	
4995	32	0	1	1	0	
4996	80	0	1	1	1	
4997	51	1	0	0	1	
4998	76	1	0	1	0	
4999	33	0	1	0	0	

	EXPOSURE_TO_POLLUTION	LONG_TERM_ILLNESS	ENERGY_LEVEL	IMMUNE_WEAKNESS	\
0	1	0	57.831178	0	
1	1	1	47.694835	1	
2	0	0	59.577435	0	
3	1	0	59.785767	0	
4	1	1	59.733941	0	
…	…	…	…	…	
4995	0	1	60.700696	1	
4996	1	1	50.751741	0	
4997	0	0	61.063496	1	
4998	0	0	48.662872	0	
4999	1	1	58.245188	0	

	BREATHING_ISSUE	ALCOHOL_CONSUMPTION	THROAT_DISCOMFORT	\
0	0	1	1	
1	1	0	1	
2	1	1	0	
3	1	0	1	
4	1	0	1	
…	…	…	…	
4995	1	1	1	
4996	1	1	1	
4997	0	0	0	

```
4998                1                    0                  1
4999                1                    1                  1

      OXYGEN_SATURATION  CHEST_TIGHTNESS  FAMILY_HISTORY  \
0             95.977287                1               0
1             97.184483                0               0
2             94.974939                0               0
3             95.187900                0               0
4             93.503008                0               0
…                   …                …               …
4995          94.012495                0               1
4996          94.394968                0               0
4997          98.108901                0               0
4998          95.577773                1               0
4999          94.206934                1               0

      SMOKING_FAMILY_HISTORY  STRESS_IMMUNE
0                          0              0
1                          0              0
2                          0              0
3                          0              0
4                          0              0
…                        …              …
4995                       1              0
4996                       0              0
4997                       0              1
4998                       0              0
4999                       0              0

[5000 rows x 17 columns]
```

[17]:
```
#For feature selection → Look at predictor vs. response correlation
# For multicollinearity → Look at predictor vs. predictor correlation
#  Use both approaches for better model performance!
A=data_cleaned.drop(columns=["PULMONARY_DISEASE"]).corr()
A
```

[17]:
```
                            AGE    GENDER   SMOKING  FINGER_DISCOLORATION  \
AGE                    1.000000 -0.004262 -0.030163             -0.012559
GENDER                -0.004262  1.000000  0.010182             -0.020919
SMOKING               -0.030163  0.010182  1.000000              0.005892
FINGER_DISCOLORATION  -0.012559 -0.020919  0.005892              1.000000
MENTAL_STRESS         -0.027137 -0.014236  0.008839              0.001116
EXPOSURE_TO_POLLUTION -0.004834 -0.024890 -0.008753              0.009729
LONG_TERM_ILLNESS      0.020401  0.017220  0.009048             -0.021592
ENERGY_LEVEL          -0.006921 -0.006845  0.018924             -0.003429
IMMUNE_WEAKNESS       -0.023072  0.020156  0.007399              0.000193
```

BREATHING_ISSUE	0.000708	0.006202	0.007500	0.010217
ALCOHOL_CONSUMPTION	-0.001551	-0.015577	0.013131	-0.015992
THROAT_DISCOMFORT	0.032412	-0.031971	-0.002220	0.017094
OXYGEN_SATURATION	-0.001354	-0.006655	0.014124	0.011930
CHEST_TIGHTNESS	-0.005792	0.027683	0.010220	0.000497
FAMILY_HISTORY	0.024816	0.037199	0.013309	-0.012645
SMOKING_FAMILY_HISTORY	0.009668	0.037498	0.358182	-0.012390
STRESS_IMMUNE	-0.027076	-0.000253	-0.001446	0.005963

	MENTAL_STRESS	EXPOSURE_TO_POLLUTION	\
AGE	-0.027137	-0.004834	
GENDER	-0.014236	-0.024890	
SMOKING	0.008839	-0.008753	
FINGER_DISCOLORATION	0.001116	0.009729	
MENTAL_STRESS	1.000000	0.003466	
EXPOSURE_TO_POLLUTION	0.003466	1.000000	
LONG_TERM_ILLNESS	0.027975	-0.016238	
ENERGY_LEVEL	0.002126	-0.011571	
IMMUNE_WEAKNESS	-0.014420	0.003616	
BREATHING_ISSUE	0.008755	0.006977	
ALCOHOL_CONSUMPTION	-0.005022	0.005995	
THROAT_DISCOMFORT	0.003985	0.012767	
OXYGEN_SATURATION	0.014609	-0.000082	
CHEST_TIGHTNESS	0.005719	0.004455	
FAMILY_HISTORY	0.012624	0.003798	
SMOKING_FAMILY_HISTORY	0.006377	-0.005284	
STRESS_IMMUNE	0.475476	-0.006655	

	LONG_TERM_ILLNESS	ENERGY_LEVEL	IMMUNE_WEAKNESS	\
AGE	0.020401	-0.006921	-0.023072	
GENDER	0.017220	-0.006845	0.020156	
SMOKING	0.009048	0.018924	0.007399	
FINGER_DISCOLORATION	-0.021592	-0.003429	0.000193	
MENTAL_STRESS	0.027975	0.002126	-0.014420	
EXPOSURE_TO_POLLUTION	-0.016238	-0.011571	0.003616	
LONG_TERM_ILLNESS	1.000000	0.020933	0.000840	
ENERGY_LEVEL	0.020933	1.000000	-0.006170	
IMMUNE_WEAKNESS	0.000840	-0.006170	1.000000	
BREATHING_ISSUE	0.019481	-0.000905	0.019460	
ALCOHOL_CONSUMPTION	0.000149	0.003187	0.004970	
THROAT_DISCOMFORT	-0.015141	0.002492	-0.026960	
OXYGEN_SATURATION	0.015301	0.008761	0.005463	
CHEST_TIGHTNESS	0.003359	-0.005799	0.000347	
FAMILY_HISTORY	0.007240	0.004743	0.008242	
SMOKING_FAMILY_HISTORY	0.004016	0.011229	0.012493	
STRESS_IMMUNE	0.012592	-0.012590	0.637578	

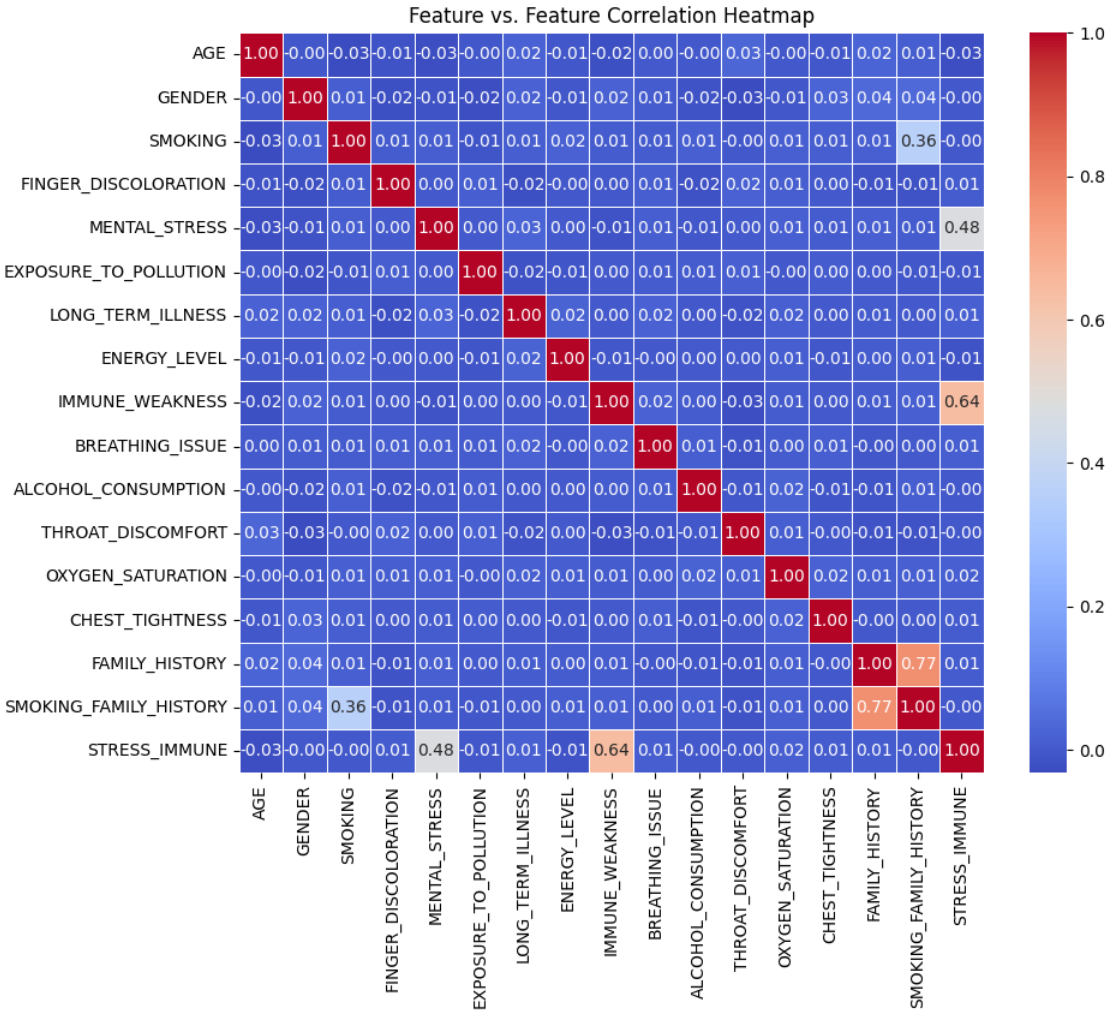
	BREATHING_ISSUE	ALCOHOL_CONSUMPTION	\
AGE	0.000708	-0.001551	
GENDER	0.006202	-0.015577	
SMOKING	0.007500	0.013131	
FINGER_DISCOLORATION	0.010217	-0.015992	
MENTAL_STRESS	0.008755	-0.005022	
EXPOSURE_TO_POLLUTION	0.006977	0.005995	
LONG_TERM_ILLNESS	0.019481	0.000149	
ENERGY_LEVEL	-0.000905	0.003187	
IMMUNE_WEAKNESS	0.019460	0.004970	
BREATHING_ISSUE	1.000000	0.012023	
ALCOHOL_CONSUMPTION	0.012023	1.000000	
THROAT_DISCOMFORT	-0.012205	-0.005932	
OXYGEN_SATURATION	0.004543	0.020444	
CHEST_TIGHTNESS	0.010624	-0.008250	
FAMILY_HISTORY	-0.004146	-0.009554	
SMOKING_FAMILY_HISTORY	0.003219	0.012158	
STRESS_IMMUNE	0.010057	-0.000207	

	THROAT_DISCOMFORT	OXYGEN_SATURATION	CHEST_TIGHTNESS	\
AGE	0.032412	-0.001354	-0.005792	
GENDER	-0.031971	-0.006655	0.027683	
SMOKING	-0.002220	0.014124	0.010220	
FINGER_DISCOLORATION	0.017094	0.011930	0.000497	
MENTAL_STRESS	0.003985	0.014609	0.005719	
EXPOSURE_TO_POLLUTION	0.012767	-0.000082	0.004455	
LONG_TERM_ILLNESS	-0.015141	0.015301	0.003359	
ENERGY_LEVEL	0.002492	0.008761	-0.005799	
IMMUNE_WEAKNESS	-0.026960	0.005463	0.000347	
BREATHING_ISSUE	-0.012205	0.004543	0.010624	
ALCOHOL_CONSUMPTION	-0.005932	0.020444	-0.008250	
THROAT_DISCOMFORT	1.000000	0.010196	-0.001507	
OXYGEN_SATURATION	0.010196	1.000000	0.019221	
CHEST_TIGHTNESS	-0.001507	0.019221	1.000000	
FAMILY_HISTORY	-0.012893	0.012119	-0.001161	
SMOKING_FAMILY_HISTORY	-0.014233	0.014289	0.003433	
STRESS_IMMUNE	-0.000764	0.015975	0.006593	

	FAMILY_HISTORY	SMOKING_FAMILY_HISTORY	STRESS_IMMUNE
AGE	0.024816	0.009668	-0.027076
GENDER	0.037199	0.037498	-0.000253
SMOKING	0.013309	0.358182	-0.001446
FINGER_DISCOLORATION	-0.012645	-0.012390	0.005963
MENTAL_STRESS	0.012624	0.006377	0.475476
EXPOSURE_TO_POLLUTION	0.003798	-0.005284	-0.006655
LONG_TERM_ILLNESS	0.007240	0.004016	0.012592
ENERGY_LEVEL	0.004743	0.011229	-0.012590

IMMUNE_WEAKNESS	0.008242	0.012493	0.637578
BREATHING_ISSUE	-0.004146	0.003219	0.010057
ALCOHOL_CONSUMPTION	-0.009554	0.012158	-0.000207
THROAT_DISCOMFORT	-0.012893	-0.014233	-0.000764
OXYGEN_SATURATION	0.012119	0.014289	0.015975
CHEST_TIGHTNESS	-0.001161	0.003433	0.006593
FAMILY_HISTORY	1.000000	0.769997	0.006116
SMOKING_FAMILY_HISTORY	0.769997	1.000000	-0.003405
STRESS_IMMUNE	0.006116	-0.003405	1.000000

[18]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(A, annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5)
plt.title("Feature vs. Feature Correlation Heatmap")
plt.show()
```



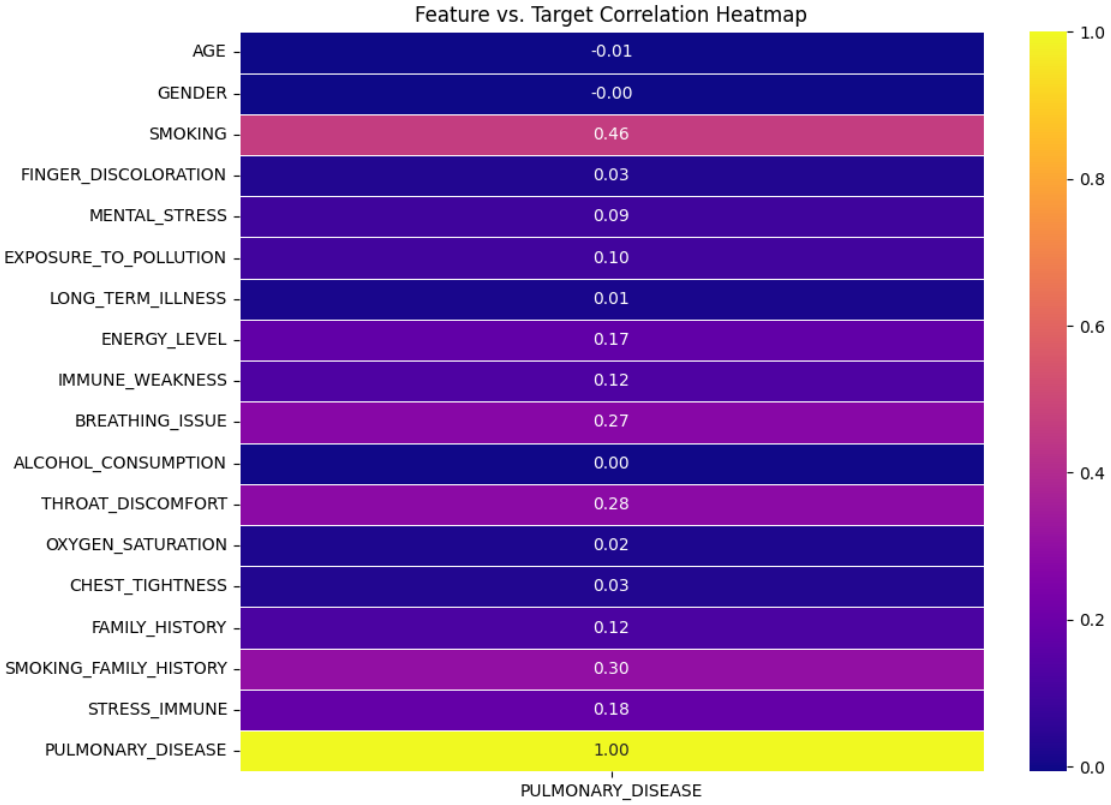
[19]:
```
# Select the upper triangle of the correlation matrix
C=data_cleaned.drop(columns=["PULMONARY_DISEASE"]).corr().abs()
upper_tri = C.where(np.triu(np.ones(C.shape), k=1).astype(bool))
upper_tri = upper_tri.fillna(0)



# Find all feature pairs with correlation > 0.75
high_corr_pairs = []
threshold = 0.75  # Define correlation threshold

for col in upper_tri.columns:
    for row in upper_tri.index:
        if upper_tri.loc[row, col] > threshold:
            high_corr_pairs.append((row, col, upper_tri.loc[row, col]))

# Convert to DataFrame for better readability
high_corr_df = pd.DataFrame(high_corr_pairs, columns=["Feature 1", "Feature 2",␣
 ↪"Correlation"])
print(high_corr_df)
```

```
       Feature 1               Feature 2  Correlation
0  FAMILY_HISTORY  SMOKING_FAMILY_HISTORY     0.769997
```

[20]:
```
B=corr_with_target = data_cleaned.corr()['PULMONARY_DISEASE']
B
```

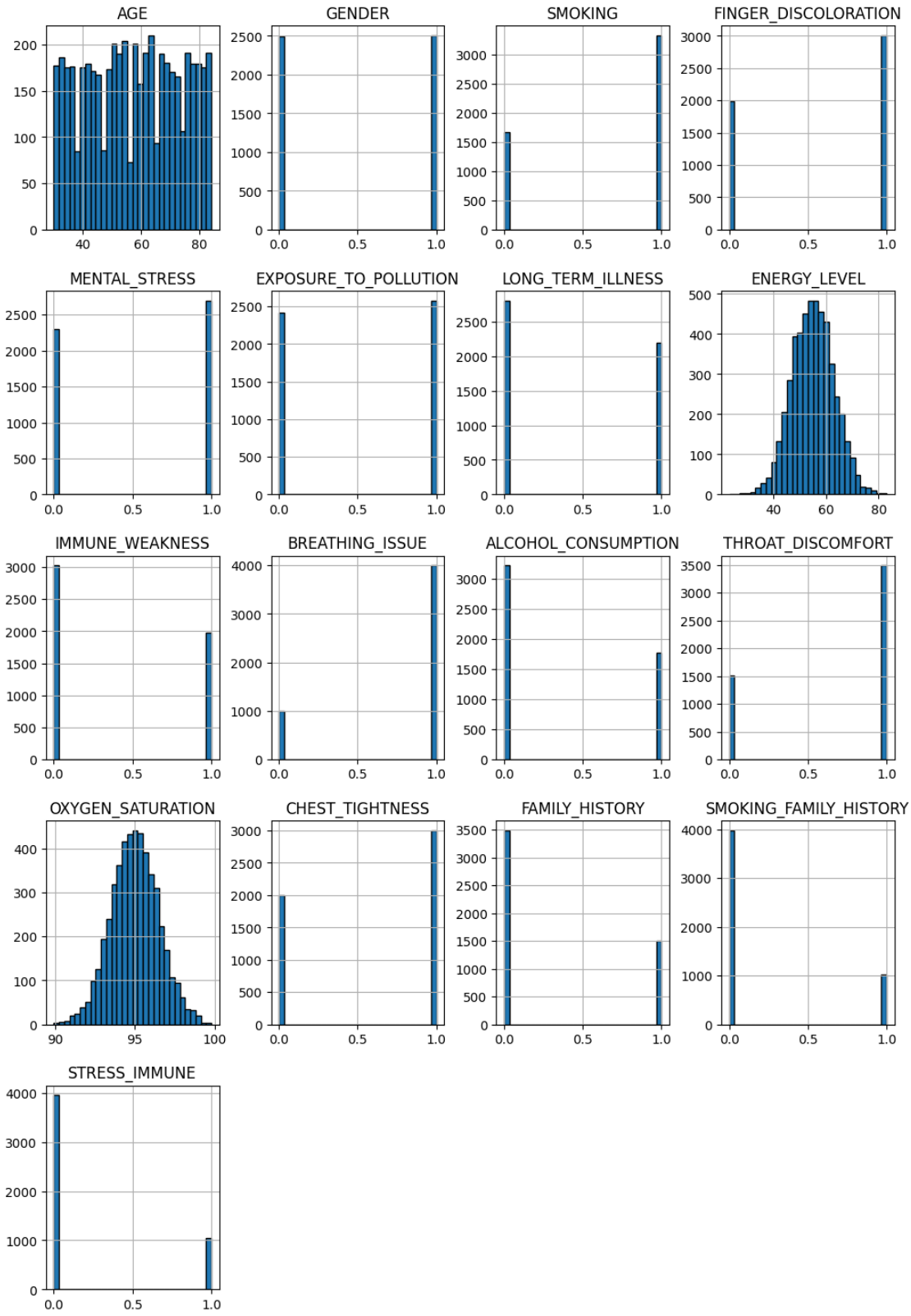
[20]:
```
AGE                      -0.006489
GENDER                   -0.004025
SMOKING                   0.461467
FINGER_DISCOLORATION      0.026066
MENTAL_STRESS             0.089367
EXPOSURE_TO_POLLUTION     0.095222
LONG_TERM_ILLNESS         0.012589
ENERGY_LEVEL              0.171479
IMMUNE_WEAKNESS           0.124736
BREATHING_ISSUE           0.270464
ALCOHOL_CONSUMPTION       0.000421
THROAT_DISCOMFORT         0.283545
OXYGEN_SATURATION         0.018570
CHEST_TIGHTNESS           0.026244
FAMILY_HISTORY            0.117255
SMOKING_FAMILY_HISTORY    0.302478
STRESS_IMMUNE             0.181053
PULMONARY_DISEASE         1.000000
Name: PULMONARY_DISEASE, dtype: float64
```

[21]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(B.to_frame(), annot=True, fmt=".2f", cmap="plasma", linewidths=0.5)
plt.title("Feature vs. Target Correlation Heatmap")
plt.show()
```

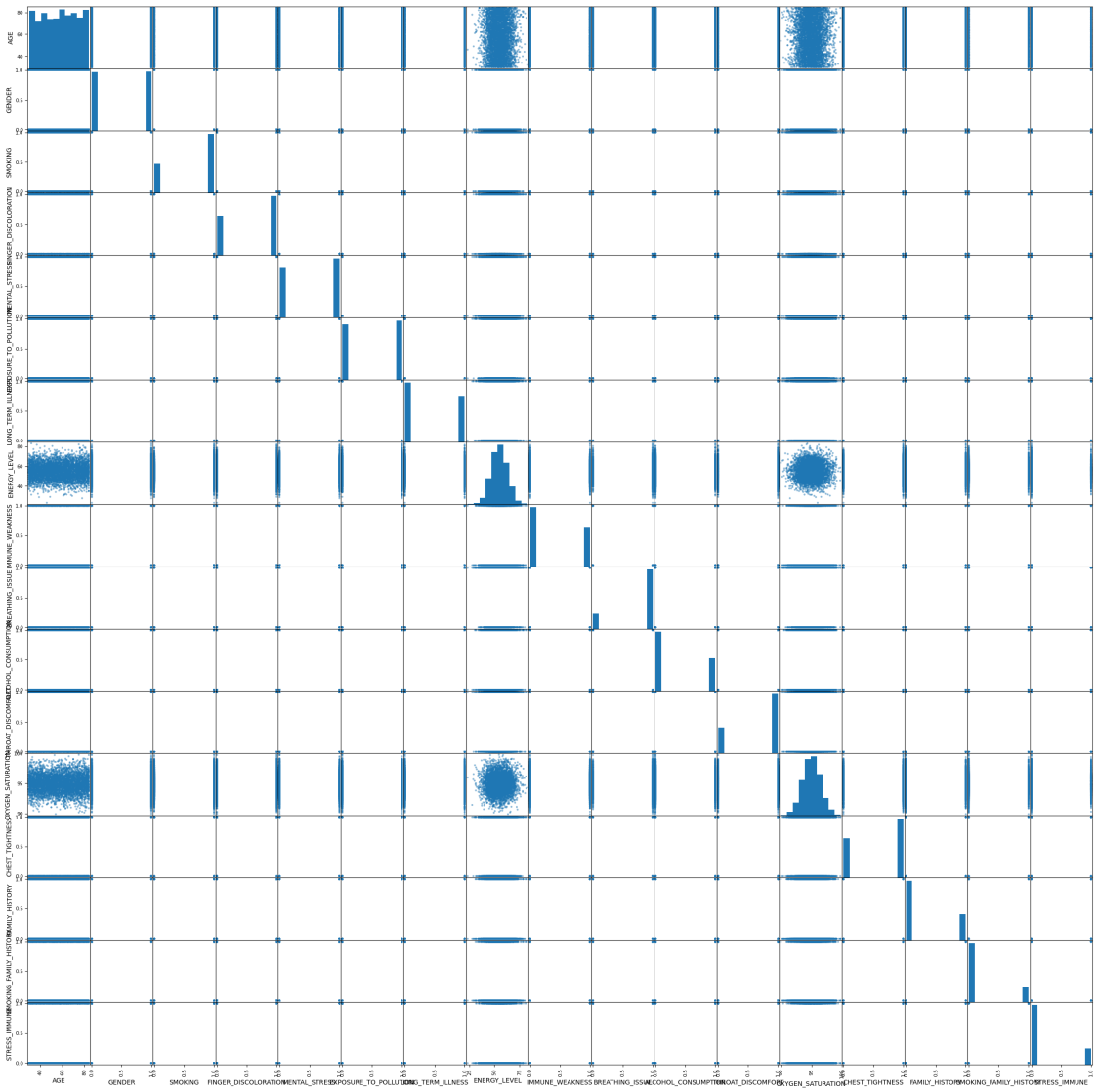


[22]:
```
# Plot histogram for each column
data_cleaned.drop(columns=["PULMONARY_DISEASE"]).hist(figsize=(12, 18),␣
↪bins=30, edgecolor='black')
plt.suptitle("Histograms of All Numerical Feature  Columns:", fontsize=20)
plt.show()
```

Histograms of All Numerical Feature Columns:

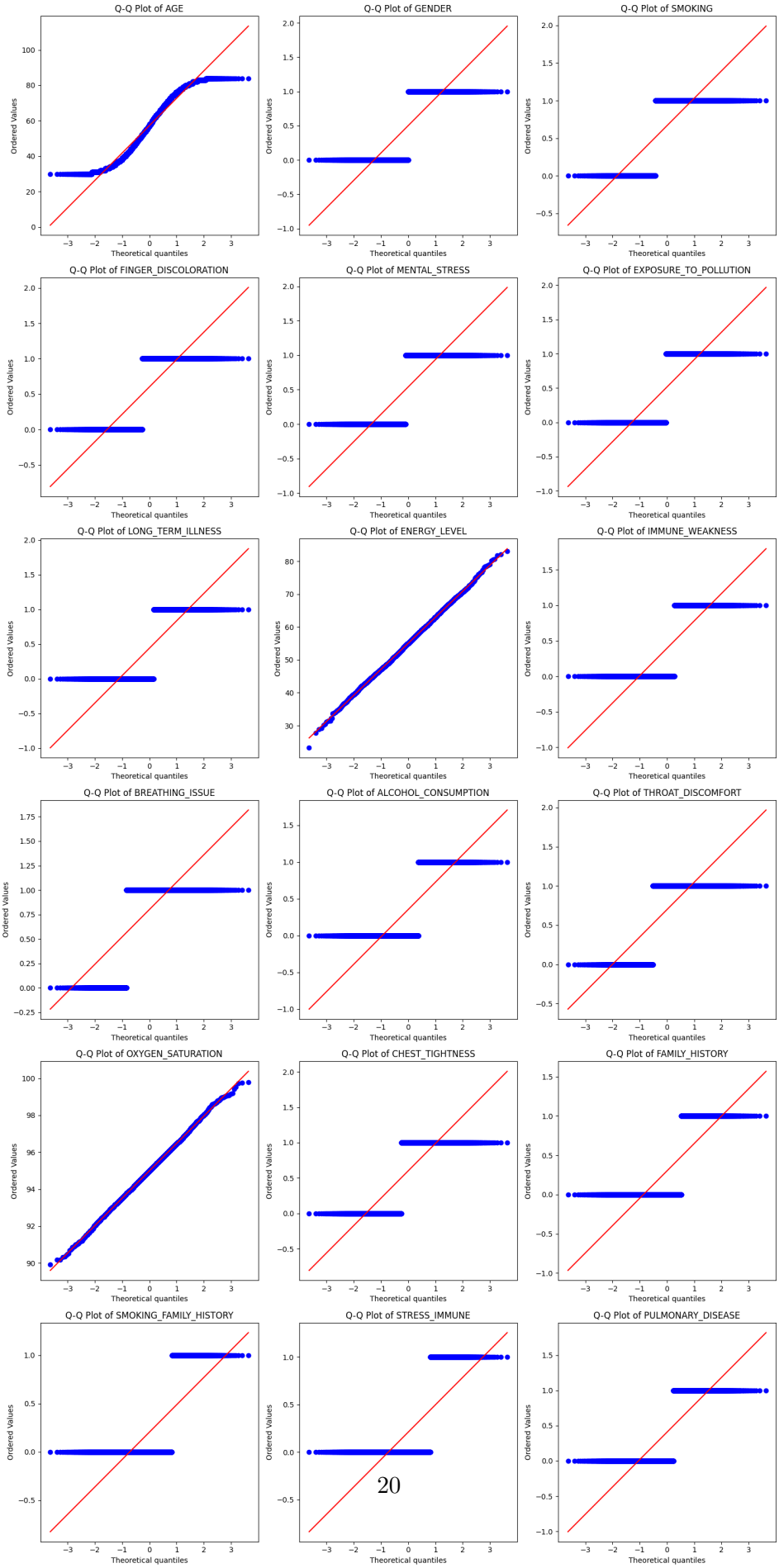


[23]:
```
scatter_matrix(data_cleaned.drop(columns=["PULMONARY_DISEASE"]), figsize=(30,␣
↪30))
plt.show()
```



[24]:
```
#Q-Q Plot :
# Number of columns
num_cols = len(data_cleaned.columns)

# Create subplots dynamically (3 columns per row)
rows = (num_cols // 3) + (num_cols % 3 > 0)  # Ensures enough rows
```

```
fig, axes = plt.subplots(rows, 3, figsize=(15, 5 * rows))

# Flatten axes for easy iteration
axes = axes.flatten()

# Generate Q-Q plots for each column
for i, col in enumerate(data_cleaned.columns):
    stats.probplot(data_cleaned[col], dist="norm", plot=axes[i])
    axes[i].set_title(f"Q-Q Plot of {col}")

# Hide any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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


Q-Q Plot of AGE
Q-Q Plot of GENDER
Q-Q Plot of SMOKING
Q-Q Plot of FINGER_DISCOLORATION
Q-Q Plot of MENTAL_STRESS
Q-Q Plot of EXPOSURE_TO_POLLUTION
Q-Q Plot of LONG_TERM_ILLNESS
Q-Q Plot of ENERGY_LEVEL
Q-Q Plot of IMMUNE_WEAKNESS
Q-Q Plot of BREATHING_ISSUE
Q-Q Plot of ALCOHOL_CONSUMPTION
Q-Q Plot of THROAT_DISCOMFORT
Q-Q Plot of OXYGEN_SATURATION
Q-Q Plot of CHEST_TIGHTNESS
Q-Q Plot of FAMILY_HISTORY
Q-Q Plot of SMOKING_FAMILY_HISTORY
Q-Q Plot of STRESS_IMMUNE
Q-Q Plot of PULMONARY_DISEASE
Ordered Values
Theoretical quantiles

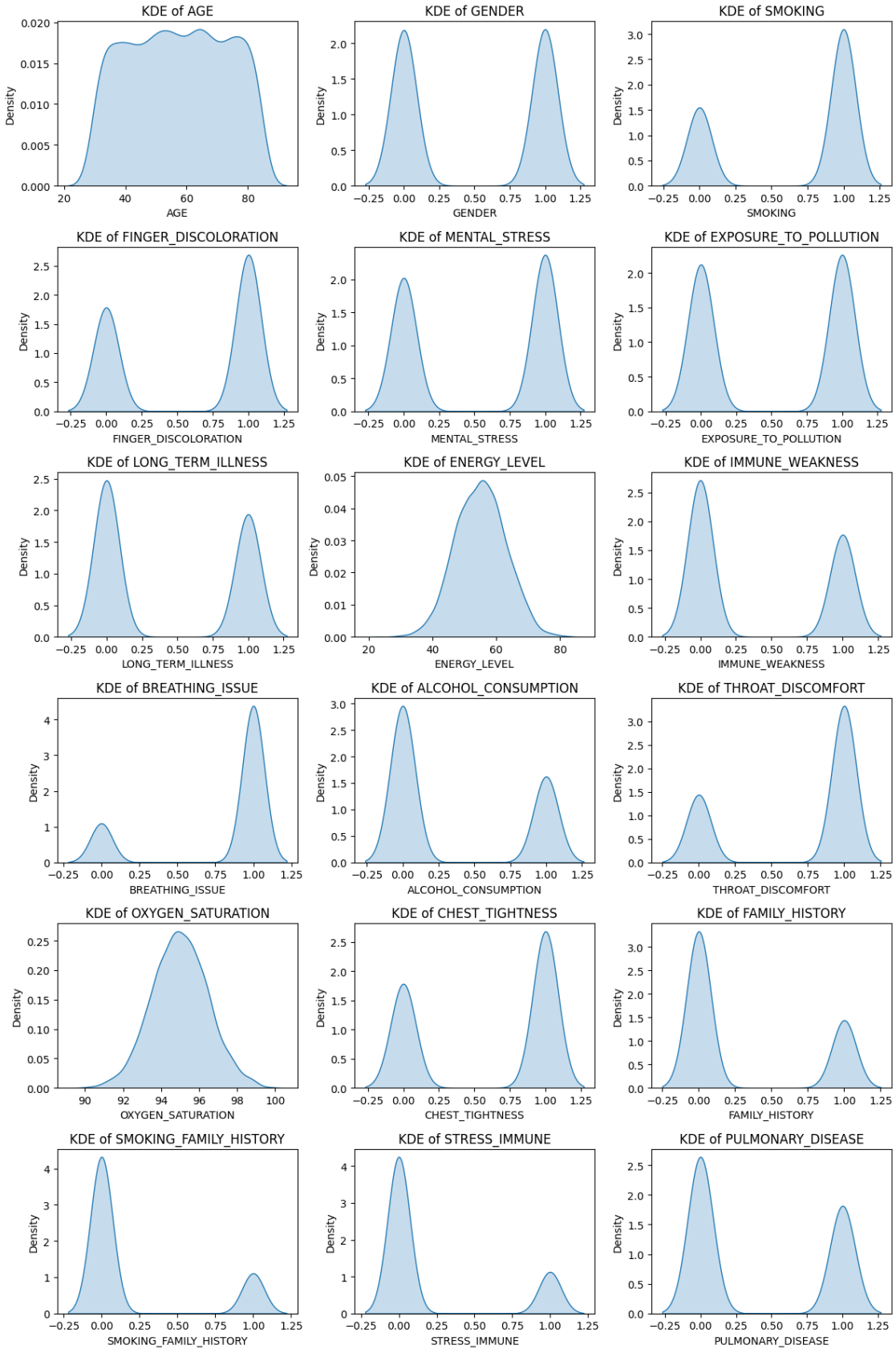
[25]:

```
# Set up the grid for subplots
num_cols = data_cleaned.shape[1]  # Total number of numerical columns
cols_per_row = 3  # Number of plots per row
rows = (num_cols // cols_per_row) + (num_cols % cols_per_row > 0)  # Calculate␣
↪required rows

# Create subplots
fig, axes = plt.subplots(rows, cols_per_row, figsize=(12, rows * 3))  # Adjust␣
↪size as needed
axes = axes.flatten()  # Flatten in case of single row

# Plot KDE for each numerical feature
for i, col in enumerate(data_cleaned.columns):
    sns.kdeplot(data_cleaned[col], fill=True, ax=axes[i])
    axes[i].set_title(f"KDE of {col}")

# Remove extra subplots if any
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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

```
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
```

```
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/usr/local/lib/python3.10/dist-packages/seaborn/_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version.
```

```
Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


KDE of AGE
Density
AGE
KDE of GENDER
Density
GENDER
KDE of SMOKING
Density
SMOKING
KDE of FINGER_DISCOLORATION
Density
FINGER_DISCOLORATION
KDE of MENTAL_STRESS
Density
MENTAL_STRESS
KDE of EXPOSURE_TO_POLLUTION
Density
EXPOSURE_TO_POLLUTION
KDE of LONG_TERM_ILLNESS
Density
LONG_TERM_ILLNESS
KDE of ENERGY_LEVEL
Density
ENERGY_LEVEL
KDE of IMMUNE_WEAKNESS
Density
IMMUNE_WEAKNESS
KDE of BREATHING_ISSUE
Density
BREATHING_ISSUE
KDE of ALCOHOL_CONSUMPTION
Density
ALCOHOL_CONSUMPTION
KDE of THROAT_DISCOMFORT
Density
THROAT_DISCOMFORT
KDE of OXYGEN_SATURATION
Density
OXYGEN_SATURATION
KDE of CHEST_TIGHTNESS
Density
CHEST_TIGHTNESS
KDE of FAMILY_HISTORY
Density
FAMILY_HISTORY
KDE of SMOKING_FAMILY_HISTORY
Density
SMOKING_FAMILY_HISTORY
KDE of STRESS_IMMUNE
Density
STRESS_IMMUNE
KDE of PULMONARY_DISEASE
Density
PULMONARY_DISEASE

[26]:
```
#Skewness  0 → Normal
#Skewness < -0.5 or > 0.5 → Skewed
print(Indep.skew())
```

```
AGE                        -0.018267
GENDER                     -0.004801
SMOKING                    -0.706046
FINGER_DISCOLORATION       -0.413479
MENTAL_STRESS              -0.159755
EXPOSURE_TO_POLLUTION      -0.064052
LONG_TERM_ILLNESS           0.245092
ENERGY_LEVEL                0.025655
IMMUNE_WEAKNESS             0.430564
BREATHING_ISSUE            -1.503580
ALCOHOL_CONSUMPTION         0.609880
THROAT_DISCOMFORT          -0.863802
OXYGEN_SATURATION           0.014458
CHEST_TIGHTNESS            -0.410924
FAMILY_HISTORY              0.863802
SMOKING_FAMILY_HISTORY      1.469537
STRESS_IMMUNE               1.427374
dtype: float64
```

[27]:
```
#Kurtosis  3 → Normal
#Kurtosis > 3 → Heavy-tailed (Leptokurtic)
#Kurtosis < 3 → Light-tailed (Platykurtic)

print(Indep.kurtosis())
```

```
AGE                        -1.176472
GENDER                     -2.000777
SMOKING                    -1.502100
FINGER_DISCOLORATION       -1.829767
MENTAL_STRESS              -1.975269
EXPOSURE_TO_POLLUTION      -1.996696
LONG_TERM_ILLNESS          -1.940706
ENERGY_LEVEL               -0.029961
IMMUNE_WEAKNESS            -1.815341
BREATHING_ISSUE             0.260856
ALCOHOL_CONSUMPTION        -1.628698
THROAT_DISCOMFORT          -1.254347
OXYGEN_SATURATION           0.006510
CHEST_TIGHTNESS            -1.831874
FAMILY_HISTORY             -1.254347
SMOKING_FAMILY_HISTORY      0.159602
STRESS_IMMUNE               0.037412
```

```
dtype: float64
```

[28]:
```python
#p > 0.05 → Data is normally distributed
# p < 0.05 → Data is not normal
# Shapiro-Wilk Test (Good for small datasets, <5000 samples)
for col in Indep.columns:
    stat, p = shapiro(Indep[col])
    print(f"{col}: p-value = {p}")

# Kolmogorov-Smirnov Test (For large datasets)

#for col in df.columns:
 #   stat, p = kstest(df[col], 'norm')
  #  print(f"{col}: p-value = {p}")

# Anderson-Darling Test

# for col in df.columns:
# result = anderson(df[col])
#  print(f"{col}: Test Statistic = {result.statistic}")
```

```
AGE: p-value = 2.053958819928301e-36
GENDER: p-value = 2.1203127046344666e-73
SMOKING: p-value = 1.175614842133481e-75
FINGER_DISCOLORATION: p-value = 3.0516417545995523e-74
MENTAL_STRESS: p-value = 1.5690906654998812e-73
EXPOSURE_TO_POLLUTION: p-value = 2.0200033959357485e-73
LONG_TERM_ILLNESS: p-value = 1.0507116633416471e-73
ENERGY_LEVEL: p-value = 0.4602502640824031
IMMUNE_WEAKNESS: p-value = 2.612572704833724e-74
BREATHING_ISSUE: p-value = 1.3282803419106409e-80
ALCOHOL_CONSUMPTION: p-value = 3.8806966520122937e-75
THROAT_DISCOMFORT: p-value = 1.3867281181651882e-76
OXYGEN_SATURATION: p-value = 0.3485019562497307
CHEST_TIGHTNESS: p-value = 3.1219986301900124e-74
FAMILY_HISTORY: p-value = 1.3867281181652674e-76
SMOKING_FAMILY_HISTORY: p-value = 2.1345532441827824e-80
STRESS_IMMUNE: p-value = 3.864344427478164e-80
```

[29]:
```python
# Create dictionary mapping column names to their positions
col_positions = {col: idx for idx, col in enumerate(data_cleaned.columns)}
print(col_positions)

# Define X (features) and y (target)
x = data_cleaned.iloc[:, :-1].values  # Convert to NumPy array
y = data_cleaned.iloc[:, -1].values   # Convert to NumPy array

print(x)
```

```
print(y)

# Split data into train/test sets
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.3, stratify=y, random_state=42)  # Stratify preserves␣
 ↪class balance

# Convert back to DataFrame (important for indexing)
x_train_df = pd.DataFrame(x_train, columns=data_cleaned.columns[:-1])
x_test_df = pd.DataFrame(x_test, columns=data_cleaned.columns[:-1])
```

```
{'AGE': 0, 'GENDER': 1, 'SMOKING': 2, 'FINGER_DISCOLORATION': 3,
'MENTAL_STRESS': 4, 'EXPOSURE_TO_POLLUTION': 5, 'LONG_TERM_ILLNESS': 6,
'ENERGY_LEVEL': 7, 'IMMUNE_WEAKNESS': 8, 'BREATHING_ISSUE': 9,
'ALCOHOL_CONSUMPTION': 10, 'THROAT_DISCOMFORT': 11, 'OXYGEN_SATURATION': 12,
'CHEST_TIGHTNESS': 13, 'FAMILY_HISTORY': 14, 'SMOKING_FAMILY_HISTORY': 15,
'STRESS_IMMUNE': 16, 'PULMONARY_DISEASE': 17}
[[68.  1.  1. … 0.  0.  0.]
 [81.  1.  1. … 0.  0.  0.]
 [58.  1.  1. … 0.  0.  0.]
 …
 [51.  1.  0. … 0.  0.  1.]
 [76.  1.  0. … 0.  0.  0.]
 [33.  0.  1. … 0.  0.  0.]]
[0 1 0 … 0 0 0]
```

[30]:
```
#Verify split  # Convert y_train and y_test to Pandas Series to use␣
 ↪value_counts()
y_train = pd.Series(y_train)
y_test = pd.Series(y_test)
print("Training set class distribution:")
print(y_train.value_counts(normalize=True))

print("\nTest set class distribution:")
print(y_test.value_counts(normalize=True))
#Why Stratification Matters
#Model Training: Ensures the model sees enough minority class samples during␣
 ↪training.
#Evaluation: Test set reflects real-world class ratios, so metrics (e.g.,␣
 ↪precision, recall) are reliable.
# For extreme imbalance like 95-5% or less :
#from sklearn.model_selection import StratifiedKFold
#skf = StratifiedKFold(n_splits=5)
```

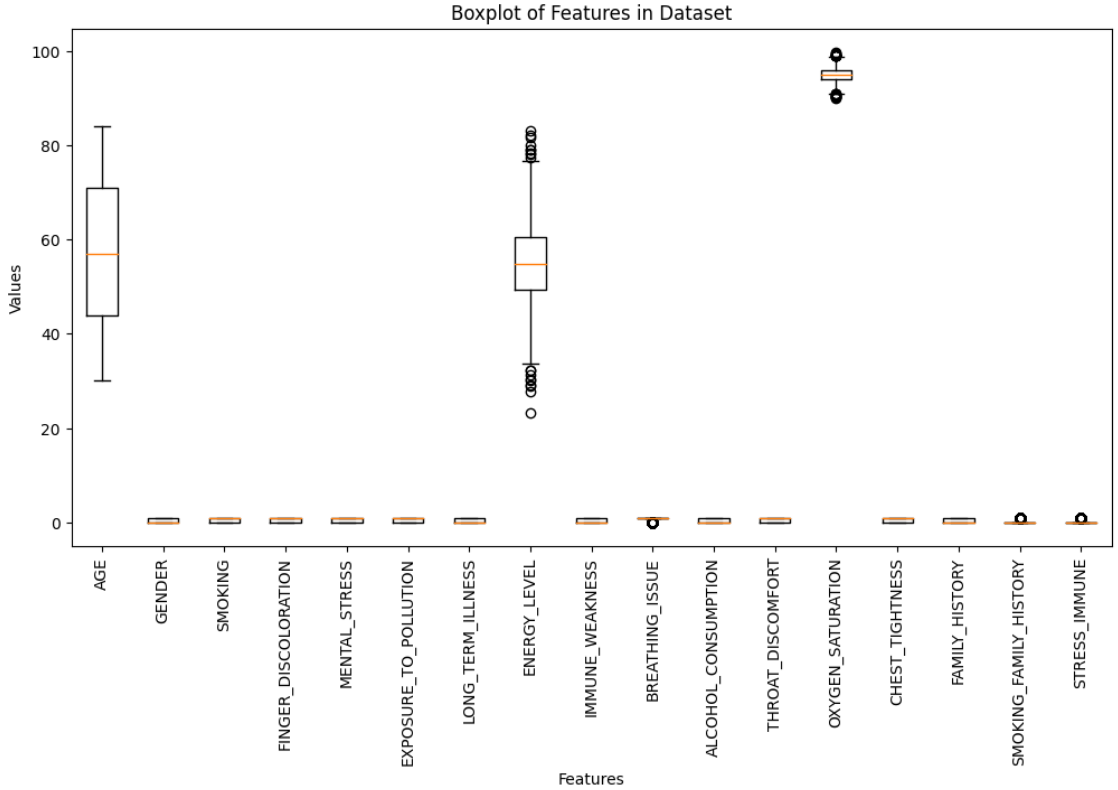
```
Training set class distribution:
0    0.592571
1    0.407429
Name: proportion, dtype: float64
```

```
Test set class distribution:
0    0.592667
1    0.407333
Name: proportion, dtype: float64
```

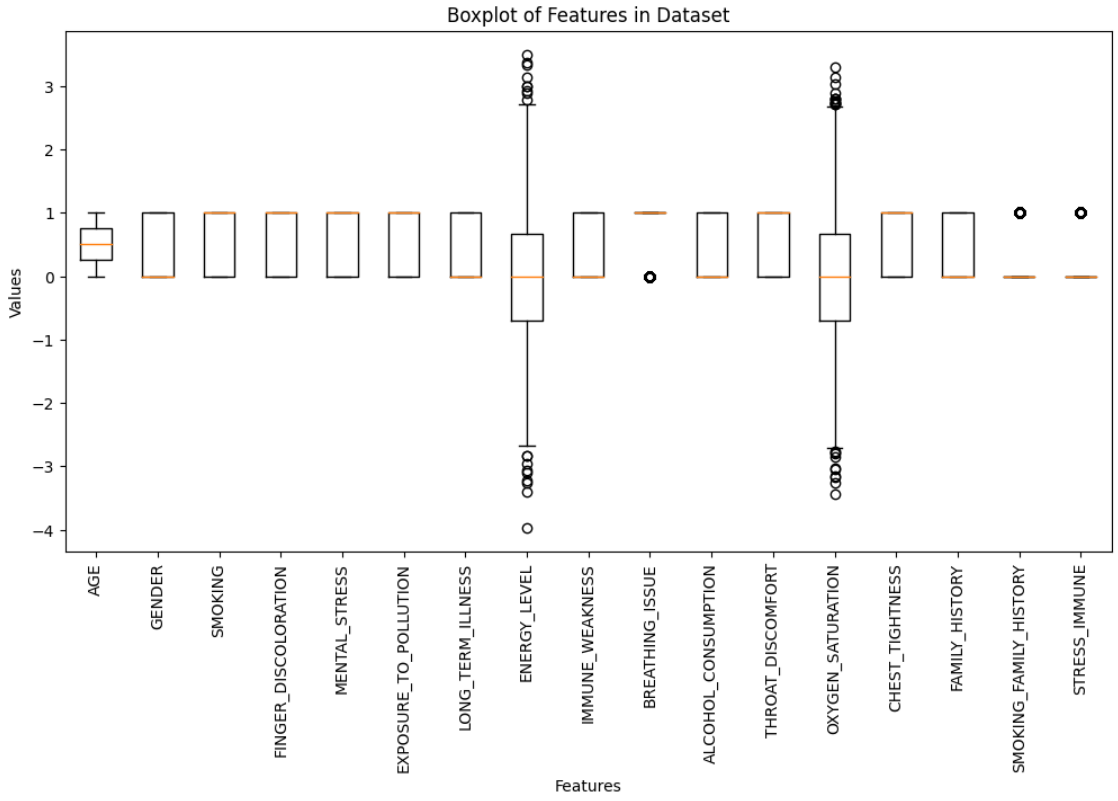
[31]:
```
#Fix: Always split before preprocessing!
# Create copies for different scaling methods
x_train_df1 = x_train_df.copy()
x_test_df1 = x_test_df.copy()
    # Define column groups
gaussian_cols = [7,12]  # Indices of Gaussian-distributed continuous features
non_gaussian_cols = [0]# Indices of Non-Gaussian continuous features
robust_scale_cols =[7,12] # Indices of Gaussian with heavy Outliers features
non_gaussian_cols1= [0,7,12] # Assuming all non Gaussian
# WHICH SCALING SHOULD WE DO ?
# CASE 1
# Standardize Gaussian features
scaler_gaussian = StandardScaler()
x_train_df1.iloc[:, gaussian_cols] = scaler_gaussian.fit_transform(x_train_df1.
 ↪iloc[:, gaussian_cols])
x_test_df1.iloc[:, gaussian_cols] = scaler_gaussian.transform(x_test_df1.iloc[:
 ↪, gaussian_cols])
# MinMax Scale Non-Gaussian features
scaler_non_gaussian = MinMaxScaler()
x_train_df1.iloc[:, non_gaussian_cols] = scaler_non_gaussian.
 ↪fit_transform(x_train_df1.iloc[:, non_gaussian_cols])
x_test_df1.iloc[:, non_gaussian_cols] = scaler_non_gaussian.
 ↪transform(x_test_df1.iloc[:, non_gaussian_cols])

# CASE 2
x_train_df2 = x_train_df.copy()
x_test_df2 = x_test_df.copy()
# Standardize Robust features
robust_scaler = RobustScaler()
x_train_df2.iloc[:, robust_scale_cols] = robust_scaler.
 ↪fit_transform(x_train_df2.iloc[:, robust_scale_cols])
x_test_df2.iloc[:, robust_scale_cols] = robust_scaler.transform(x_test_df2.
 ↪iloc[:, robust_scale_cols])
# MinMax Scale Non-Gaussian features
scaler_non_gaussian = MinMaxScaler()
x_train_df2.iloc[:, non_gaussian_cols] = scaler_non_gaussian.
 ↪fit_transform(x_train_df2.iloc[:, non_gaussian_cols])
x_test_df2.iloc[:, non_gaussian_cols] = scaler_non_gaussian.
 ↪transform(x_test_df2.iloc[:, non_gaussian_cols])
```

```
# CASE 3
# MinMax Scale Non-Gaussian features
x_train_df3 = x_train_df.copy()
x_test_df3 = x_test_df.copy()

scaler_non_gaussian1 = MinMaxScaler()
x_train_df3.iloc[:, non_gaussian_cols1] = scaler_non_gaussian1.
 ↪fit_transform(x_train_df3.iloc[:, non_gaussian_cols1])
x_test_df3.iloc[:, non_gaussian_cols1] = scaler_non_gaussian1.
 ↪transform(x_test_df3.iloc[:, non_gaussian_cols1])
```

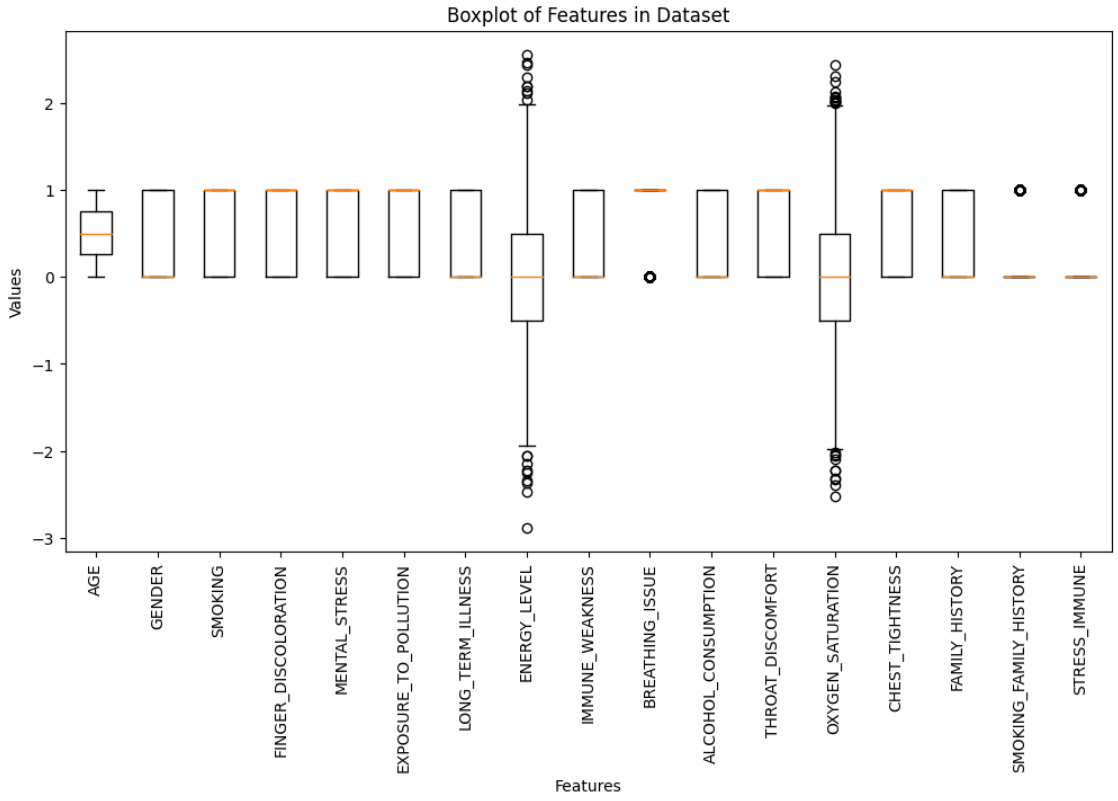
[32]:
```
plt.figure(figsize=(12, 6))  # Set figure size
plt.boxplot(x_train_df.values, labels=x_train_df.columns, vert=True)  # Boxplot␣
 ↪for all features
plt.xticks(rotation=90)  # Rotate feature names for readability
plt.title("Boxplot of Features in Dataset")
plt.xlabel("Features")
plt.ylabel("Values")
plt.show()
```



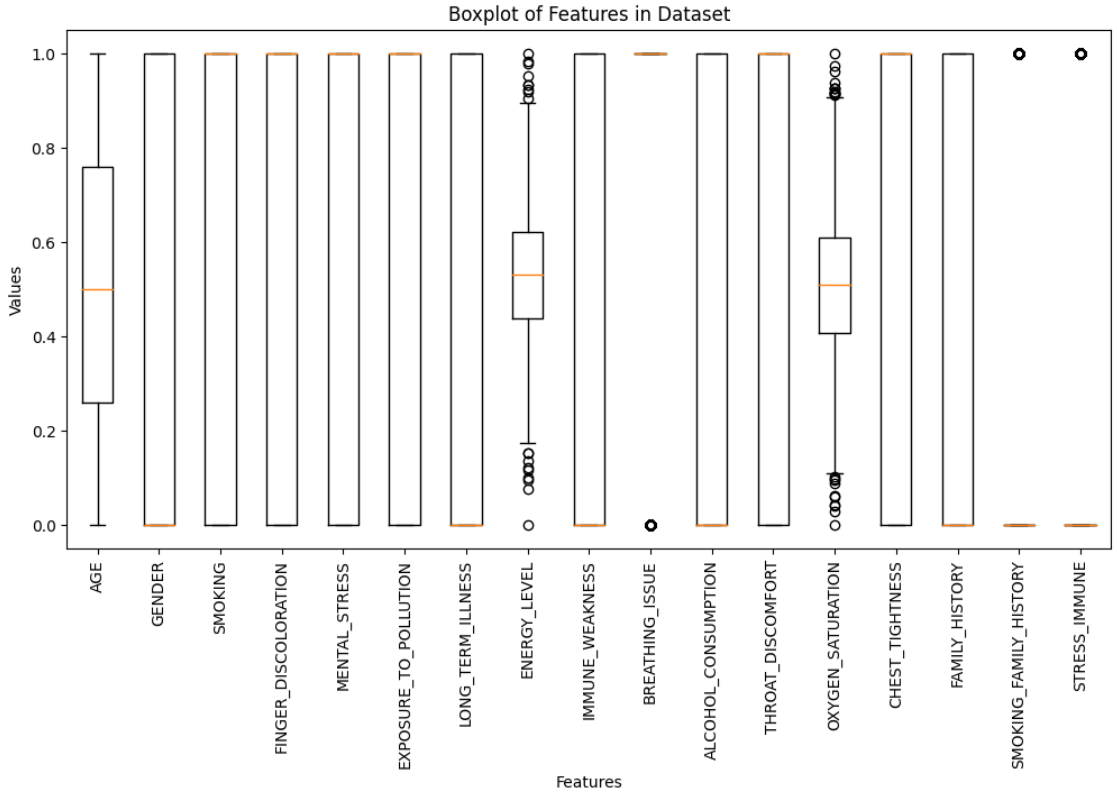
[33]:
```
plt.figure(figsize=(12, 6))  # Set figure size
plt.boxplot(x_train_df1.values, labels=x_train_df1.columns, vert=True)  #␣
 ↪Boxplot for all features
plt.xticks(rotation=90)  # Rotate feature names for readability
plt.title("Boxplot of Features in Dataset")
plt.xlabel("Features")
plt.ylabel("Values")
plt.show()
```



[34]:
```
plt.figure(figsize=(12, 6))  # Set figure size
plt.boxplot(x_train_df2.values, labels=x_train_df2.columns, vert=True)  #␣
 ↪Boxplot for all features
plt.xticks(rotation=90)  # Rotate feature names for readability
plt.title("Boxplot of Features in Dataset")
plt.xlabel("Features")
plt.ylabel("Values")
plt.show()
```



[35]:
```
plt.figure(figsize=(12, 6))  # Set figure size
plt.boxplot(x_train_df3.values, labels=x_train_df3.columns, vert=True)  #␣
↪Boxplot for all features
plt.xticks(rotation=90)  # Rotate feature names for readability
plt.title("Boxplot of Features in Dataset")
plt.xlabel("Features")
plt.ylabel("Values")
plt.show()
```



We only worked with case 1 as its the most appropiate in this case.

[36]:
```
# Function to calculate VIF
def calculate_vif(df):
    vif_data = pd.DataFrame()
    vif_data["Feature"] = df.columns
    vif_data["VIF"] = [variance_inflation_factor(df.values, i) for i in␣
 ↪range(df.shape[1])]
    return vif_data

# Load your dataset (assuming 'data_cleaned' exists and target variable is␣
 ↪removed)
X_0 = x_train_df # Independent variables

# Iteratively remove features with high VIF
threshold = 10  # VIF threshold
step = 1  # Step counter
dropped_features = []
while True:
    print(f"\nStep {step}: Calculating VIF...\n")
    vif_data = calculate_vif(X_0)
    print(vif_data)  # Print current VIF values
```

```
    max_vif = vif_data["VIF"].max()  # Find the max VIF value

    if max_vif < threshold:  # Stop when all VIFs are below threshold
        print("\nAll remaining features have VIF  10. Stopping iteration.\n")
        break

    # Find the feature with the highest VIF and drop it
    feature_to_drop = vif_data.loc[vif_data["VIF"] == max_vif, "Feature"].
↪values[0]
    print(f"\nDropping '{feature_to_drop}' with VIF: {max_vif}\n")

    X_0 =X_0.drop(columns=[feature_to_drop])  # Drop the feature
    step += 1  # Increment step counter
    dropped_features.append(feature_to_drop)  # Store the dropped feature

# Now drop the same features from x_test
x_test_vif = x_test_df.drop(columns=dropped_features)

# Final VIF Data
print("\nFinal VIF Data:")
print(calculate_vif(X_0))
x_train_vif=X_0
print(X_0.columns)
print(x_test_vif.columns)
```

```
Step 1: Calculating VIF…

                   Feature        VIF
0                      AGE  14.315481
1                   GENDER   2.000457
2                  SMOKING   4.285793
3     FINGER_DISCOLORATION   2.519714
4            MENTAL_STRESS   3.679460
5    EXPOSURE_TO_POLLUTION   2.117468
6        LONG_TERM_ILLNESS   1.788105
7             ENERGY_LEVEL  47.852352
8          IMMUNE_WEAKNESS   3.668080
9          BREATHING_ISSUE   5.000296
10     ALCOHOL_CONSUMPTION   1.566112
11       THROAT_DISCOMFORT   3.389690
12       OXYGEN_SATURATION  80.390618
13         CHEST_TIGHTNESS   2.492612
14          FAMILY_HISTORY   4.420489
15  SMOKING_FAMILY_HISTORY   4.460968
16           STRESS_IMMUNE   3.621725
```

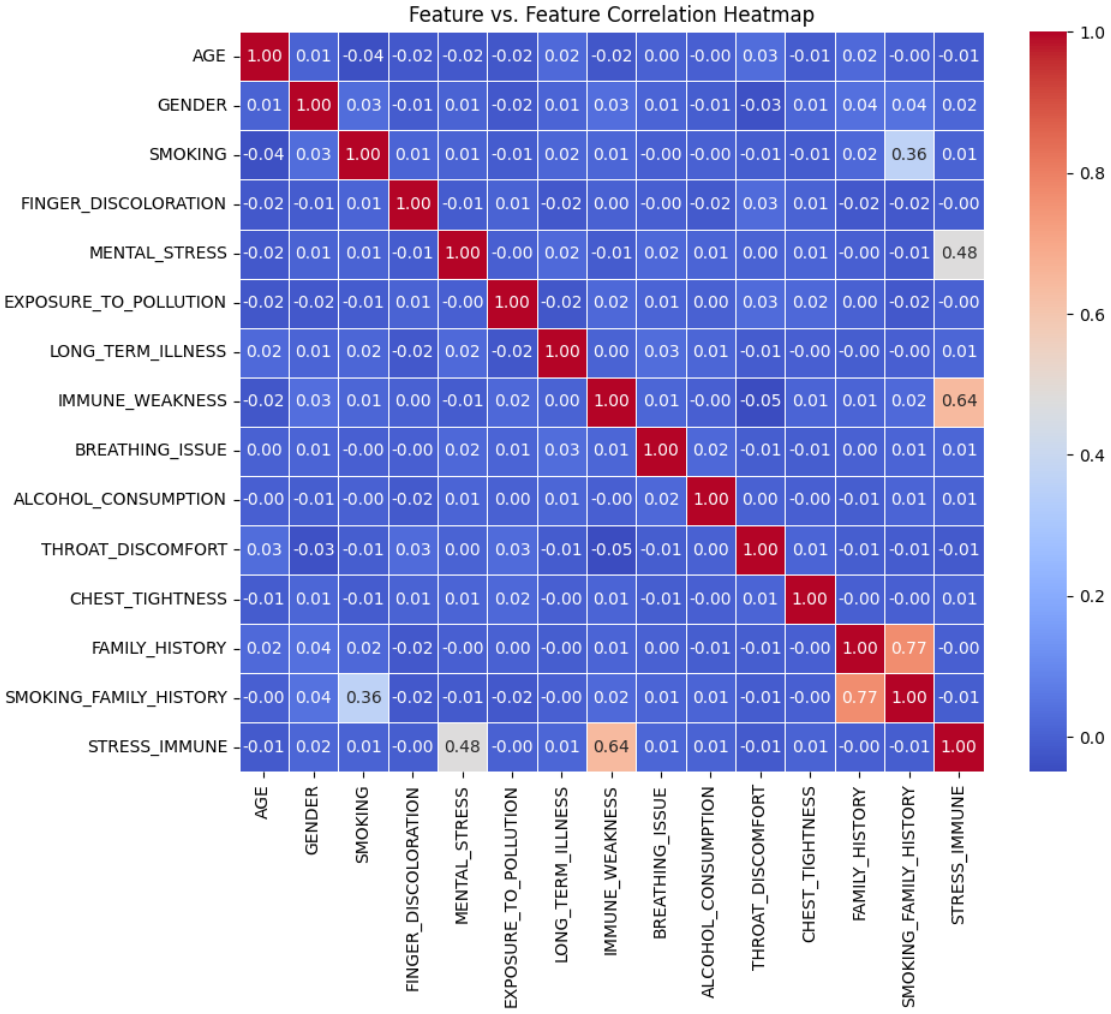
```
Dropping 'OXYGEN_SATURATION' with VIF: 80.39061765931686


Step 2: Calculating VIF…

                   Feature        VIF
0                      AGE  11.723225
1                   GENDER   1.979068
2                  SMOKING   4.141227
3     FINGER_DISCOLORATION   2.457261
4            MENTAL_STRESS   3.603951
5    EXPOSURE_TO_POLLUTION   2.090313
6        LONG_TERM_ILLNESS   1.778177
7             ENERGY_LEVEL  19.536945
8          IMMUNE_WEAKNESS   3.607817
9          BREATHING_ISSUE   4.779513
10     ALCOHOL_CONSUMPTION   1.553709
11       THROAT_DISCOMFORT   3.291157
12         CHEST_TIGHTNESS   2.446298
13          FAMILY_HISTORY   4.357874
14  SMOKING_FAMILY_HISTORY   4.416740
15           STRESS_IMMUNE   3.596843

Dropping 'ENERGY_LEVEL' with VIF: 19.53694452361641


Step 3: Calculating VIF…

                   Feature       VIF
0                      AGE  8.382613
1                   GENDER  1.951539
2                  SMOKING  3.897903
3     FINGER_DISCOLORATION  2.396371
4            MENTAL_STRESS  3.450873
5    EXPOSURE_TO_POLLUTION  2.048915
6        LONG_TERM_ILLNESS  1.758488
7          IMMUNE_WEAKNESS  3.484225
8          BREATHING_ISSUE  4.431082
9      ALCOHOL_CONSUMPTION  1.537615
10       THROAT_DISCOMFORT  3.168429
11         CHEST_TIGHTNESS  2.380929
12          FAMILY_HISTORY  4.280375
13  SMOKING_FAMILY_HISTORY  4.360489
14           STRESS_IMMUNE  3.521864

All remaining features have VIF  10. Stopping iteration.
```

```
Final VIF Data:
                    Feature       VIF
0                       AGE  8.382613
1                    GENDER  1.951539
2                   SMOKING  3.897903
3     FINGER_DISCOLORATION  2.396371
4            MENTAL_STRESS  3.450873
5    EXPOSURE_TO_POLLUTION  2.048915
6        LONG_TERM_ILLNESS  1.758488
7          IMMUNE_WEAKNESS  3.484225
8          BREATHING_ISSUE  4.431082
9      ALCOHOL_CONSUMPTION  1.537615
10       THROAT_DISCOMFORT  3.168429
11         CHEST_TIGHTNESS  2.380929
12          FAMILY_HISTORY  4.280375
13  SMOKING_FAMILY_HISTORY  4.360489
14           STRESS_IMMUNE  3.521864
Index(['AGE', 'GENDER', 'SMOKING', 'FINGER_DISCOLORATION', 'MENTAL_STRESS',
       'EXPOSURE_TO_POLLUTION', 'LONG_TERM_ILLNESS', 'IMMUNE_WEAKNESS',
       'BREATHING_ISSUE', 'ALCOHOL_CONSUMPTION', 'THROAT_DISCOMFORT',
       'CHEST_TIGHTNESS', 'FAMILY_HISTORY', 'SMOKING_FAMILY_HISTORY',
       'STRESS_IMMUNE'],
      dtype='object')
Index(['AGE', 'GENDER', 'SMOKING', 'FINGER_DISCOLORATION', 'MENTAL_STRESS',
       'EXPOSURE_TO_POLLUTION', 'LONG_TERM_ILLNESS', 'IMMUNE_WEAKNESS',
       'BREATHING_ISSUE', 'ALCOHOL_CONSUMPTION', 'THROAT_DISCOMFORT',
       'CHEST_TIGHTNESS', 'FAMILY_HISTORY', 'SMOKING_FAMILY_HISTORY',
       'STRESS_IMMUNE'],
      dtype='object')
```

[37]:
```
# Convert to DataFrame
x_train_vif_df = pd.DataFrame(x_train_vif)
```

If the p-value in the Box-Tidwell test is less than 0.05, it means that the feature violates the linearity assumption of logistic regression.

[38]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(x_train_vif_df.corr(), annot=True, fmt=".2f", cmap="coolwarm",␣
 ↪linewidths=0.5)
plt.title("Feature vs. Feature Correlation Heatmap")
plt.show()
```



[39]:
```
# After dropping high correlated factors, we need to check if it impacts␣
 ↪performance of the model
x_train_vif_1=  x_train_vif.drop('FAMILY_HISTORY', axis=1).
 ↪reset_index(drop=True)
x_train_vif_2=  x_train_vif.drop('SMOKING_FAMILY_HISTORY', axis=1).
 ↪reset_index(drop=True)
x_train_vif_3=  x_train_vif.drop(['FAMILY_HISTORY','SMOKING_FAMILY_HISTORY'],␣
 ↪axis=1).reset_index(drop=True)
x_train_vif_4=  x_train_vif.drop('IMMUNE_WEAKNESS', axis=1).
 ↪reset_index(drop=True)
x_train_vif_5=  x_train_vif.drop('STRESS_IMMUNE', axis=1).reset_index(drop=True)
x_train_vif_6=  x_train_vif.drop(['STRESS_IMMUNE','IMMUNE_WEAKNESS'], axis=1).
 ↪reset_index(drop=True)
x_train_vif_7=  x_train_vif.drop(['FAMILY_HISTORY','IMMUNE_WEAKNESS'], axis=1).
 ↪reset_index(drop=True)
```

```
x_train_vif_8=  x_train_vif.drop(['STRESS_IMMUNE','SMOKING_FAMILY_HISTORY'],␣
 ↪axis=1).reset_index(drop=True)
x_train_vif_9=  x_train_vif.
 ↪drop(['STRESS_IMMUNE','SMOKING_FAMILY_HISTORY','FAMILY_HISTORY','IMMUNE_WEAKNESS'],␣
 ↪axis=1).reset_index(drop=True)


x_test_vif_1=  x_test_vif.drop('FAMILY_HISTORY', axis=1).reset_index(drop=True)
x_test_vif_2=  x_test_vif.drop('SMOKING_FAMILY_HISTORY', axis=1).
 ↪reset_index(drop=True)
x_test_vif_3=  x_test_vif.drop(['FAMILY_HISTORY','SMOKING_FAMILY_HISTORY'],␣
 ↪axis=1).reset_index(drop=True).reset_index(drop=True).reset_index(drop=True).
 ↪reset_index(drop=True).reset_index(drop=True).reset_index(drop=True).
 ↪reset_index(drop=True)
x_test_vif_4=  x_test_vif.drop('IMMUNE_WEAKNESS', axis=1).
 ↪reset_index(drop=True).reset_index(drop=True).reset_index(drop=True).
 ↪reset_index(drop=True).reset_index(drop=True).reset_index(drop=True)
x_test_vif_5=  x_test_vif.drop('STRESS_IMMUNE', axis=1).reset_index(drop=True).
 ↪reset_index(drop=True).reset_index(drop=True).reset_index(drop=True).
 ↪reset_index(drop=True)
x_test_vif_6=  x_test_vif.drop(['STRESS_IMMUNE','IMMUNE_WEAKNESS'], axis=1).
 ↪reset_index(drop=True).reset_index(drop=True).reset_index(drop=True).
 ↪reset_index(drop=True)
x_test_vif_7=  x_test_vif.drop(['FAMILY_HISTORY','IMMUNE_WEAKNESS'], axis=1).
 ↪reset_index(drop=True).reset_index(drop=True).reset_index(drop=True)
x_test_vif_8=  x_test_vif.drop(['STRESS_IMMUNE','SMOKING_FAMILY_HISTORY'],␣
 ↪axis=1).reset_index(drop=True).reset_index(drop=True)
x_test_vif_9=  x_test_vif.
 ↪drop(['STRESS_IMMUNE','SMOKING_FAMILY_HISTORY','FAMILY_HISTORY','IMMUNE_WEAKNESS'],␣
 ↪axis=1).reset_index(drop=True)
print(x_train_vif.index) # Before dropping

print(x_train_vif_1.index)
print(x_train_vif_2.index)
```

```
RangeIndex(start=0, stop=3500, step=1)
RangeIndex(start=0, stop=3500, step=1)
RangeIndex(start=0, stop=3500, step=1)
```

[40]:
```
# If the interaction term (log_X1) is significant (p < 0.05), the linearity␣
 ↪assumption is violated.
# Define your datasets and their corresponding target variables
datasets = {
    "D0": (x_train_vif, y_train),
    "D1": (x_train_vif_1, y_train),
    "D2": (x_train_vif_2, y_train),
    "D3": (x_train_vif_3, y_train),
```

```
    "D4": (x_train_vif_4, y_train),
    "D5": (x_train_vif_5, y_train),
    "D6": (x_train_vif_6, y_train),
    "D7": (x_train_vif_7, y_train),
    "D8": (x_train_vif_8, y_train),
    "D9": (x_train_vif_9, y_train)
}

# Define the feature to check (modify as needed)
box_tidwell_feature0 = "AGE"  # Change this to any column name


# Function to perform Box-Tidwell test
def box_tidwell_test(X, y, dataset_name, feature):
    X = X.copy()  # Avoid modifying original dataset
    if feature not in X.columns:
        print(f"Feature '{feature}' not found in {dataset_name}, skipping...")
        return

    X[f"log_{feature}"] = np.log(X[feature] + 1) * X[feature]  # Apply␣
 ↪transformation
    X = sm.add_constant(X)  # Add intercept

    try:
        logit_model = sm.Logit(y, X).fit()  # Fit logistic regression
        print(f"\n### Box-Tidwell Test Results for {dataset_name} ###")
        print(logit_model.summary())  # Print results
    except Exception as e:
        print(f"Error in {dataset_name}: {e}")

# Run for each dataset
for name, (X, y) in datasets.items():
    box_tidwell_test(X, y, name, box_tidwell_feature0)
```

```
Optimization terminated successfully.
         Current function value: 0.379932
         Iterations 7

### Box-Tidwell Test Results for D0 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3483
Method:                           MLE   Df Model:                           16
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                 0.4379
Time:                        19:50:49   Log-Likelihood:                -1329.8
converged:                       True   LL-Null:                       -2365.7
```

```
Covariance Type:            nonrobust   LLR p-value:                    0.000
==============================================================================
==========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const                     -9.2576      1.307     -7.085      0.000     -11.819
-6.697
AGE                        0.0891      0.120      0.745      0.457      -0.146
0.324
GENDER                    -0.0152      0.098     -0.156      0.876      -0.207
0.176
SMOKING                    3.2454      0.158     20.519      0.000       2.935
3.555
FINGER_DISCOLORATION       0.1661      0.100      1.665      0.096      -0.029
0.362
MENTAL_STRESS             -0.1607      0.128     -1.255      0.209      -0.412
0.090
EXPOSURE_TO_POLLUTION      0.8149      0.100      8.128      0.000       0.618
1.011
LONG_TERM_ILLNESS          0.0859      0.098      0.873      0.383      -0.107
0.279
IMMUNE_WEAKNESS            0.0654      0.147      0.445      0.656      -0.223
0.354
BREATHING_ISSUE            2.9370      0.155     18.932      0.000       2.633
3.241
ALCOHOL_CONSUMPTION        0.0006      0.102      0.006      0.995      -0.199
0.200
THROAT_DISCOMFORT          2.4167      0.124     19.534      0.000       2.174
2.659
CHEST_TIGHTNESS            0.1754      0.100      1.759      0.079      -0.020
0.371
FAMILY_HISTORY            -0.2618      0.260     -1.009      0.313      -0.771
0.247
SMOKING_FAMILY_HISTORY     1.6964      0.289      5.865      0.000       1.130
2.263
STRESS_IMMUNE              1.7107      0.205      8.341      0.000       1.309
2.113
log_AGE                   -0.0179      0.024     -0.750      0.453      -0.065
0.029
==============================================================================
==========
Optimization terminated successfully.
         Current function value: 0.380082
         Iterations 7

### Box-Tidwell Test Results for D1 ###
```

Logit Regression Results

Dep. Variable:	y	No. Observations:	3500
Model:	Logit	Df Residuals:	3484
Method:	MLE	Df Model:	15
Date:	Wed, 02 Apr 2025	Pseudo R-squ.:	0.4377
Time:	19:50:49	Log-Likelihood:	-1330.3
converged:	True	LL-Null:	-2365.7
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-9.3156	1.305	-7.136	0.000	-11.874	-6.757
AGE	0.0879	0.120	0.734	0.463	-0.147	0.323
GENDER	-0.0168	0.098	-0.172	0.863	-0.208	0.174
SMOKING	3.3190	0.143	23.264	0.000	3.039	3.599
FINGER_DISCOLORATION	0.1657	0.100	1.661	0.097	-0.030	0.361
MENTAL_STRESS	-0.1616	0.128	-1.262	0.207	-0.413	0.089
EXPOSURE_TO_POLLUTION	0.8137	0.100	8.117	0.000	0.617	1.010
LONG_TERM_ILLNESS	0.0873	0.098	0.886	0.375	-0.106	0.280
IMMUNE_WEAKNESS	0.0643	0.147	0.437	0.662	-0.224	0.352
BREATHING_ISSUE	2.9368	0.155	18.932	0.000	2.633	3.241
ALCOHOL_CONSUMPTION	0.0031	0.102	0.031	0.975	-0.196	0.203
THROAT_DISCOMFORT	2.4167	0.124	19.535	0.000	2.174	2.659
CHEST_TIGHTNESS	0.1773	0.100	1.778	0.075	-0.018	0.373
SMOKING_FAMILY_HISTORY	1.4345	0.127	11.290	0.000	1.185	1.683
STRESS_IMMUNE	1.7113	0.205	8.345	0.000	1.309	2.113
log_AGE	-0.0176	0.024	-0.740	0.459	-0.064	0.029

```
==========
Optimization terminated successfully.
         Current function value: 0.385355
         Iterations 7

### Box-Tidwell Test Results for D2 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3484
Method:                           MLE   Df Model:                           15
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.4299
Time:                        19:50:49   Log-Likelihood:                -1348.7
converged:                       True   LL-Null:                       -2365.7
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
=========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const                     -9.5354      1.295     -7.362      0.000     -12.074
-6.997
AGE                        0.0860      0.119      0.725      0.469      -0.147
0.319
GENDER                    -0.0149      0.097     -0.154      0.878      -0.205
0.175
SMOKING                    3.7829      0.145     26.057      0.000       3.498
4.067
FINGER_DISCOLORATION       0.1543      0.099      1.560      0.119      -0.039
0.348
MENTAL_STRESS             -0.1629      0.127     -1.285      0.199      -0.411
0.086
EXPOSURE_TO_POLLUTION      0.7925      0.099      7.980      0.000       0.598
0.987
LONG_TERM_ILLNESS          0.0881      0.098      0.902      0.367      -0.103
0.280
IMMUNE_WEAKNESS            0.0767      0.145      0.528      0.598      -0.208
0.362
BREATHING_ISSUE            2.8574      0.151     18.914      0.000       2.561
3.154
ALCOHOL_CONSUMPTION        0.0214      0.101      0.212      0.832      -0.177
0.220
THROAT_DISCOMFORT          2.3490      0.120     19.532      0.000       2.113
2.585
CHEST_TIGHTNESS            0.1813      0.099      1.835      0.067      -0.012
0.375
FAMILY_HISTORY             1.0822      0.108      9.997      0.000       0.870
```

```
1.294
STRESS_IMMUNE             1.6761      0.203      8.249      0.000       1.278
2.074
log_AGE                  -0.0173      0.024     -0.731      0.465      -0.064
0.029
==============================================================================
=========
Optimization terminated successfully.
         Current function value: 0.400505
         Iterations 7

### Box-Tidwell Test Results for D3 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3485
Method:                           MLE   Df Model:                           14
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                 0.4075
Time:                        19:50:49   Log-Likelihood:                -1401.8
converged:                       True   LL-Null:                       -2365.7
Covariance Type:            nonrobust   LLR p-value:                    0.000
==============================================================================
=========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---------
const                     -9.0489      1.266     -7.145      0.000     -11.531
-6.567
AGE                        0.1043      0.116      0.897      0.370      -0.124
0.332
GENDER                     0.0210      0.095      0.221      0.825      -0.165
0.207
SMOKING                    3.6249      0.139     26.086      0.000       3.353
3.897
FINGER_DISCOLORATION       0.1313      0.097      1.354      0.176      -0.059
0.321
MENTAL_STRESS             -0.1513      0.124     -1.217      0.224      -0.395
0.092
EXPOSURE_TO_POLLUTION      0.7497      0.097      7.719      0.000       0.559
0.940
LONG_TERM_ILLNESS          0.0698      0.096      0.729      0.466      -0.118
0.257
IMMUNE_WEAKNESS            0.1355      0.143      0.948      0.343      -0.145
0.416
BREATHING_ISSUE            2.7344      0.146     18.759      0.000       2.449
3.020
ALCOHOL_CONSUMPTION        0.0285      0.099      0.287      0.774      -0.166
```

```
0.223
THROAT_DISCOMFORT         2.2263      0.115     19.307      0.000       2.000
2.452
CHEST_TIGHTNESS           0.1618      0.097      1.673      0.094      -0.028
0.351
STRESS_IMMUNE             1.5323      0.198      7.741      0.000       1.144
1.920
log_AGE                  -0.0208      0.023     -0.899      0.369      -0.066
0.025
==============================================================================
=========
Optimization terminated successfully.
         Current function value: 0.379961
         Iterations 7

### Box-Tidwell Test Results for D4 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3484
Method:                           MLE   Df Model:                           15
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.4379
Time:                        19:50:49   Log-Likelihood:                -1329.9
converged:                       True   LL-Null:                       -2365.7
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
==========
                               coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const                       -9.2428      1.306     -7.076      0.000     -11.803
-6.683
AGE                          0.0902      0.120      0.754      0.451      -0.144
0.325
GENDER                      -0.0146      0.098     -0.150      0.881      -0.206
0.177
SMOKING                      3.2466      0.158     20.529      0.000       2.937
3.557
FINGER_DISCOLORATION         0.1657      0.100      1.662      0.097      -0.030
0.361
MENTAL_STRESS               -0.1878      0.113     -1.668      0.095      -0.409
0.033
EXPOSURE_TO_POLLUTION        0.8166      0.100      8.150      0.000       0.620
1.013
LONG_TERM_ILLNESS            0.0864      0.098      0.878      0.380      -0.107
0.279
BREATHING_ISSUE              2.9382      0.155     18.946      0.000       2.634
```

```
3.242
ALCOHOL_CONSUMPTION        0.0003      0.102      0.003      0.997      -0.199
0.200
THROAT_DISCOMFORT          2.4141      0.124     19.539      0.000       2.172
2.656
CHEST_TIGHTNESS            0.1763      0.100      1.768      0.077      -0.019
0.372
FAMILY_HISTORY            -0.2610      0.260     -1.005      0.315      -0.770
0.248
SMOKING_FAMILY_HISTORY     1.6979      0.289      5.871      0.000       1.131
2.265
STRESS_IMMUNE              1.7763      0.143     12.424      0.000       1.496
2.057
log_AGE                   -0.0181      0.024     -0.759      0.448      -0.065
0.029
==============================================================================
==========
Optimization terminated successfully.
         Current function value: 0.390310
         Iterations 7

### Box-Tidwell Test Results for D5 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3484
Method:                           MLE   Df Model:                           15
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.4225
Time:                        19:50:49   Log-Likelihood:                -1366.1
converged:                       True   LL-Null:                       -2365.7
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
==========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const                      -9.2151      1.289     -7.149      0.000     -11.742
-6.689
AGE                         0.0768      0.118      0.650      0.516      -0.155
0.308
GENDER                     -0.0140      0.096     -0.145      0.884      -0.203
0.175
SMOKING                     3.1086      0.152     20.419      0.000       2.810
3.407
FINGER_DISCOLORATION        0.1612      0.098      1.639      0.101      -0.032
0.354
MENTAL_STRESS               0.5339      0.097      5.478      0.000       0.343
```

```
0.725
EXPOSURE_TO_POLLUTION      0.7737      0.099      7.835      0.000       0.580
0.967
LONG_TERM_ILLNESS          0.0699      0.097      0.720      0.471      -0.120
0.260
IMMUNE_WEAKNESS            0.9720      0.101      9.624      0.000       0.774
1.170
BREATHING_ISSUE            2.8272      0.151     18.733      0.000       2.531
3.123
ALCOHOL_CONSUMPTION       -0.0064      0.100     -0.063      0.949      -0.203
0.190
THROAT_DISCOMFORT          2.3598      0.121     19.524      0.000       2.123
2.597
CHEST_TIGHTNESS            0.1686      0.098      1.713      0.087      -0.024
0.361
FAMILY_HISTORY            -0.2647      0.254     -1.043      0.297      -0.762
0.233
SMOKING_FAMILY_HISTORY     1.6222      0.283      5.737      0.000       1.068
2.177
log_AGE                   -0.0153      0.024     -0.652      0.514      -0.061
0.031
==============================================================================
==========
Optimization terminated successfully.
         Current function value: 0.404222
         Iterations 7

### Box-Tidwell Test Results for D6 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3485
Method:                           MLE   Df Model:                           14
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.4020
Time:                        19:50:49   Log-Likelihood:                -1414.8
converged:                       True   LL-Null:                       -2365.7
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
==========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const                     -8.7235      1.266     -6.891      0.000     -11.205
-6.242
AGE                        0.0983      0.116      0.846      0.398      -0.129
0.326
GENDER                     0.0060      0.095      0.064      0.949      -0.179
```

```
0.191
SMOKING                    2.9995      0.148     20.283      0.000       2.710
3.289
FINGER_DISCOLORATION       0.1493      0.096      1.548      0.122      -0.040
0.338
MENTAL_STRESS              0.5107      0.096      5.341      0.000       0.323
0.698
EXPOSURE_TO_POLLUTION      0.7739      0.097      7.975      0.000       0.584
0.964
LONG_TERM_ILLNESS          0.0714      0.095      0.750      0.453      -0.115
0.258
BREATHING_ISSUE            2.7333      0.146     18.717      0.000       2.447
3.020
ALCOHOL_CONSUMPTION       -0.0168      0.098     -0.171      0.864      -0.209
0.176
THROAT_DISCOMFORT          2.2028      0.116     19.072      0.000       1.976
2.429
CHEST_TIGHTNESS            0.1829      0.096      1.895      0.058      -0.006
0.372
FAMILY_HISTORY            -0.2475      0.250     -0.990      0.322      -0.738
0.243
SMOKING_FAMILY_HISTORY     1.5826      0.278      5.686      0.000       1.037
2.128
log_AGE                   -0.0197      0.023     -0.852      0.394      -0.065
0.026
==============================================================================
==========
Optimization terminated successfully.
         Current function value: 0.380109
         Iterations 7

### Box-Tidwell Test Results for D7 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3485
Method:                           MLE   Df Model:                           14
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.4376
Time:                        19:50:49   Log-Likelihood:                -1330.4
converged:                       True   LL-Null:                       -2365.7
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
==========
                             coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const                     -9.3013      1.305     -7.128      0.000     -11.859
```

```
-6.744
AGE                          0.0890      0.120      0.744      0.457      -0.146
0.324
GENDER                      -0.0162      0.098     -0.166      0.868      -0.207
0.175
SMOKING                      3.3198      0.143     23.272      0.000       3.040
3.599
FINGER_DISCOLORATION         0.1653      0.100      1.658      0.097      -0.030
0.361
MENTAL_STRESS               -0.1883      0.113     -1.672      0.095      -0.409
0.032
EXPOSURE_TO_POLLUTION        0.8153      0.100      8.139      0.000       0.619
1.012
LONG_TERM_ILLNESS            0.0877      0.098      0.891      0.373      -0.105
0.281
BREATHING_ISSUE              2.9380      0.155     18.946      0.000       2.634
3.242
ALCOHOL_CONSUMPTION          0.0028      0.102      0.028      0.978      -0.197
0.202
THROAT_DISCOMFORT            2.4141      0.124     19.541      0.000       2.172
2.656
CHEST_TIGHTNESS              0.1781      0.100      1.786      0.074      -0.017
0.373
SMOKING_FAMILY_HISTORY       1.4368      0.127     11.317      0.000       1.188
1.686
STRESS_IMMUNE                1.7757      0.143     12.424      0.000       1.496
2.056
log_AGE                     -0.0179      0.024     -0.750      0.453      -0.065
0.029
==============================================================================
==========
Optimization terminated successfully.
         Current function value: 0.395489
         Iterations 7

### Box-Tidwell Test Results for D8 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3485
Method:                           MLE   Df Model:                           14
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.4149
Time:                        19:50:49   Log-Likelihood:                -1384.2
converged:                       True   LL-Null:                       -2365.7
Covariance Type:            nonrobust   LLR p-value:                    0.000
==============================================================================
=========
                             coef    std err          z      P>|z|      [0.025
```

```
0.975]
------------------------------------------------------------------------------
---------
const                     -9.4328      1.278     -7.380      0.000     -11.938
-6.928
AGE                        0.0694      0.117      0.593      0.553      -0.160
0.299
GENDER                    -0.0163      0.096     -0.171      0.864      -0.204
0.171
SMOKING                    3.6263      0.139     26.120      0.000       3.354
3.898
FINGER_DISCOLORATION       0.1505      0.097      1.544      0.123      -0.041
0.342
MENTAL_STRESS              0.5166      0.097      5.350      0.000       0.327
0.706
EXPOSURE_TO_POLLUTION      0.7518      0.098      7.689      0.000       0.560
0.943
LONG_TERM_ILLNESS          0.0734      0.096      0.762      0.446      -0.115
0.262
IMMUNE_WEAKNESS            0.9616      0.100      9.596      0.000       0.765
1.158
BREATHING_ISSUE            2.7540      0.147     18.731      0.000       2.466
3.042
ALCOHOL_CONSUMPTION        0.0151      0.100      0.152      0.879      -0.180
0.210
THROAT_DISCOMFORT          2.2945      0.118     19.504      0.000       2.064
2.525
CHEST_TIGHTNESS            0.1759      0.097      1.805      0.071      -0.015
0.367
FAMILY_HISTORY             1.0173      0.106      9.617      0.000       0.810
1.225
log_AGE                   -0.0139      0.023     -0.595      0.552      -0.060
0.032
==============================================================================
=========
Optimization terminated successfully.
         Current function value: 0.423350
         Iterations 7

### Box-Tidwell Test Results for D9 ###
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 3500
Model:                          Logit   Df Residuals:                     3487
Method:                           MLE   Df Model:                           12
Date:                Wed, 02 Apr 2025   Pseudo R-squ.:                  0.3737
Time:                        19:50:49   Log-Likelihood:                -1481.7
converged:                       True   LL-Null:                       -2365.7
```

```
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
=========
                             coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
---------
const                     -8.4439      1.228     -6.877      0.000     -10.850
-6.037
AGE                        0.1013      0.113      0.896      0.370      -0.120
0.323
GENDER                     0.0421      0.092      0.458      0.647      -0.138
0.223
SMOKING                    3.3829      0.129     26.139      0.000       3.129
3.637
FINGER_DISCOLORATION       0.1120      0.094      1.191      0.233      -0.072
0.296
MENTAL_STRESS              0.4508      0.093      4.855      0.000       0.269
0.633
EXPOSURE_TO_POLLUTION      0.7212      0.094      7.649      0.000       0.536
0.906
LONG_TERM_ILLNESS          0.0574      0.093      0.618      0.536      -0.125
0.239
BREATHING_ISSUE            2.5653      0.138     18.614      0.000       2.295
2.835
ALCOHOL_CONSUMPTION        0.0138      0.096      0.144      0.885      -0.174
0.202
THROAT_DISCOMFORT          2.0421      0.109     18.820      0.000       1.829
2.255
CHEST_TIGHTNESS            0.1743      0.094      1.856      0.063      -0.010
0.358
log_AGE                   -0.0202      0.023     -0.900      0.368      -0.064
0.024
================================================================================
=========
```

Box-Tidwell test passes for all cases so no need to worry, linearity assumption remains intact.

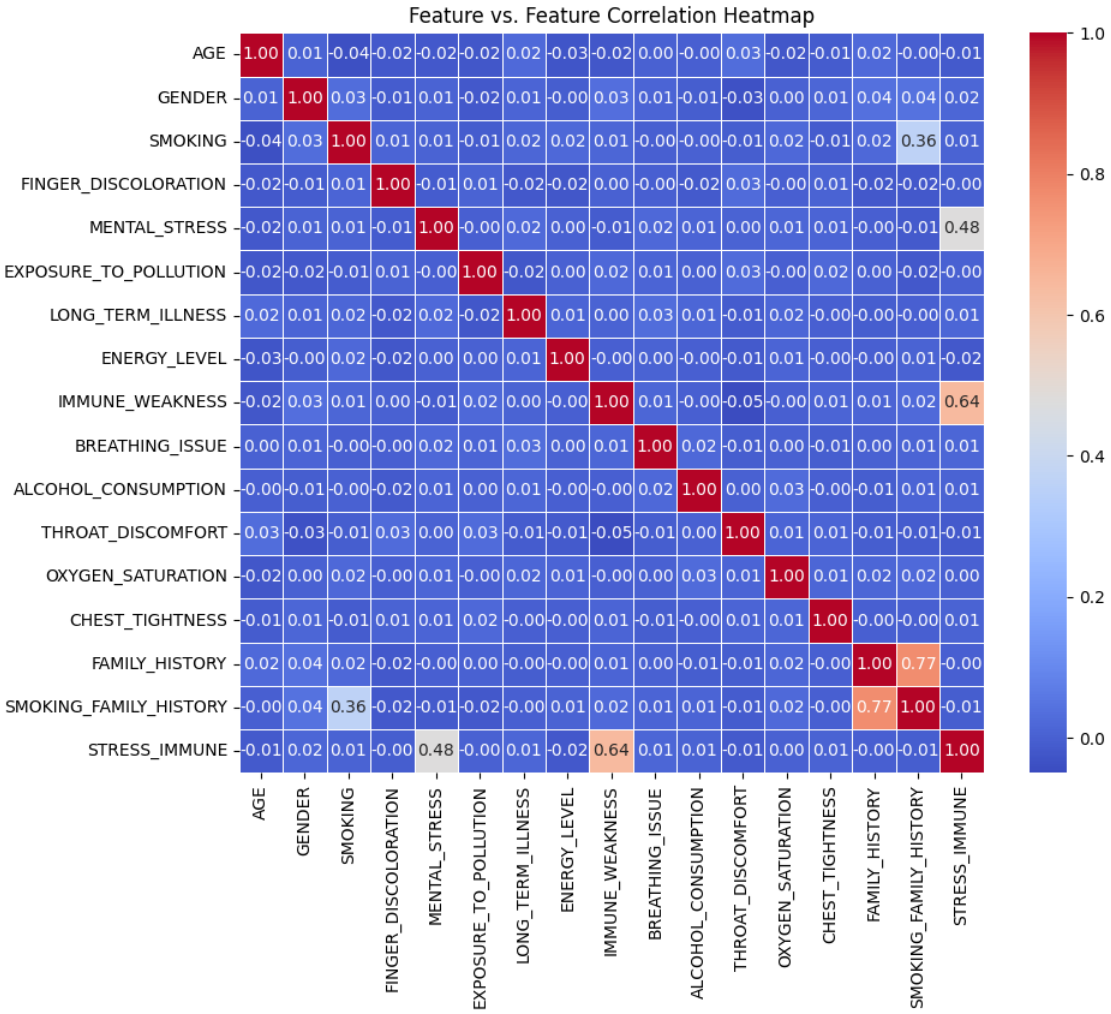
[41]:
```python
#For Standardised Data
# Function to calculate VIF
def calculate_vif(df):
    vif_data = pd.DataFrame()
    vif_data["Feature"] = df.columns
    vif_data["VIF"] = [variance_inflation_factor(df.values, i) for i in␣
 ↪range(df.shape[1])]
    return vif_data
```

```
# Load your dataset (assuming 'data_cleaned' exists and target variable is␣
 ↪removed)
X_1 = x_train_df1 # Independent variables

# Iteratively remove features with high VIF
threshold = 10  # VIF threshold
step = 1  # Step counter
dropped_features = []
while True:
    print(f"\nStep {step}: Calculating VIF...\n")
    vif_data = calculate_vif(X_1)
    print(vif_data)  # Print current VIF values

    max_vif = vif_data["VIF"].max()  # Find the max VIF value

    if max_vif < threshold:  # Stop when all VIFs are below threshold
        print("\nAll remaining features have VIF  10. Stopping iteration.\n")
        break


    # Find the feature with the highest VIF and drop it
    feature_to_drop = vif_data.loc[vif_data["VIF"] == max_vif, "Feature"].
 ↪values[0]
    print(f"\nDropping '{feature_to_drop}' with VIF: {max_vif}\n")

    X_1 =X_1.drop(columns=[feature_to_drop])  # Drop the feature
    step += 1  # Increment step counter
    dropped_features.append(feature_to_drop)  # Store the dropped feature

# Now drop the same features from x_test
x_test_vif1 = x_test_df1.drop(columns=dropped_features)

# Final VIF Data
print("\nFinal VIF Data:")
print(calculate_vif(X_1))
x_train_vif1=X_1
print(X_0.columns)
print(x_test_vif1.columns)
```

```
Step 1: Calculating VIF…

                  Feature       VIF
0                     AGE  3.467667
1                  GENDER  1.925769
2                 SMOKING  3.747396
3     FINGER_DISCOLORATION  2.349863
```

```
4            MENTAL_STRESS  3.365481
5    EXPOSURE_TO_POLLUTION  2.024781
6        LONG_TERM_ILLNESS  1.740070
7             ENERGY_LEVEL  1.002692
8          IMMUNE_WEAKNESS  3.417039
9          BREATHING_ISSUE  4.160690
10     ALCOHOL_CONSUMPTION  1.525871
11       THROAT_DISCOMFORT  3.038097
12       OXYGEN_SATURATION  1.002396
13         CHEST_TIGHTNESS  2.333594
14          FAMILY_HISTORY  4.197049
15  SMOKING_FAMILY_HISTORY  4.305037
16           STRESS_IMMUNE  3.486011

All remaining features have VIF  10. Stopping iteration.


Final VIF Data:
                   Feature       VIF
0                      AGE  3.467667
1                   GENDER  1.925769
2                  SMOKING  3.747396
3     FINGER_DISCOLORATION  2.349863
4            MENTAL_STRESS  3.365481
5    EXPOSURE_TO_POLLUTION  2.024781
6        LONG_TERM_ILLNESS  1.740070
7             ENERGY_LEVEL  1.002692
8          IMMUNE_WEAKNESS  3.417039
9          BREATHING_ISSUE  4.160690
10     ALCOHOL_CONSUMPTION  1.525871
11       THROAT_DISCOMFORT  3.038097
12       OXYGEN_SATURATION  1.002396
13         CHEST_TIGHTNESS  2.333594
14          FAMILY_HISTORY  4.197049
15  SMOKING_FAMILY_HISTORY  4.305037
16           STRESS_IMMUNE  3.486011
Index(['AGE', 'GENDER', 'SMOKING', 'FINGER_DISCOLORATION', 'MENTAL_STRESS',
       'EXPOSURE_TO_POLLUTION', 'LONG_TERM_ILLNESS', 'IMMUNE_WEAKNESS',
       'BREATHING_ISSUE', 'ALCOHOL_CONSUMPTION', 'THROAT_DISCOMFORT',
       'CHEST_TIGHTNESS', 'FAMILY_HISTORY', 'SMOKING_FAMILY_HISTORY',
       'STRESS_IMMUNE'],
      dtype='object')
Index(['AGE', 'GENDER', 'SMOKING', 'FINGER_DISCOLORATION', 'MENTAL_STRESS',
       'EXPOSURE_TO_POLLUTION', 'LONG_TERM_ILLNESS', 'ENERGY_LEVEL',
       'IMMUNE_WEAKNESS', 'BREATHING_ISSUE', 'ALCOHOL_CONSUMPTION',
       'THROAT_DISCOMFORT', 'OXYGEN_SATURATION', 'CHEST_TIGHTNESS',
       'FAMILY_HISTORY', 'SMOKING_FAMILY_HISTORY', 'STRESS_IMMUNE'],
      dtype='object')
```

[42]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(x_train_vif1.corr(), annot=True, fmt=".2f", cmap="coolwarm",␣
 ↪linewidths=0.5)
plt.title("Feature vs. Feature Correlation Heatmap")
plt.show()
```



[43]:
```
#  Step 3: Function to Train & Evaluate Model
def train_evaluate_model(model, X_train, X_test, y_train, y_test, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]

    acc = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
```

```
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_prob)


    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    r2 = r2_score(y_test, y_pred)


    print(f"\n Model: {model_name}")
    print("Accuracy:", acc)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-score:", f1)
    print("AUC-ROC:", auc)
    print("MAE:", mae)
    print("MSE:", mse)
    print("R2_SCORE:", r2)
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    return {"Model": model_name, "Accuracy": acc, "Precision": precision,␣
 ↪"Recall": recall, "F1-Score": f1, "AUC-ROC": auc,"MAE": mae,"MSE":␣
 ↪mse,"R2_SCORE": r2}

#  Step 4: Train Models & Compare
# 1 Logistic Regression Without VIF
logistic_no_vif = LogisticRegression(solver='liblinear')
results_no_vif = train_evaluate_model(logistic_no_vif, x_train, x_test,␣
 ↪y_train, y_test, "Logistic Without VIF")

# 2 Logistic Regression With VIF-filtered Features
logistic_vif = LogisticRegression(solver='liblinear')
results_vif = train_evaluate_model(logistic_vif, x_train_vif, x_test_vif,␣
 ↪y_train, y_test, "Logistic With VIF")

# 3 Logistic Regression With L1 (Lasso)
logistic_l1 = LogisticRegression(penalty='l1', solver='liblinear', C=1.0)
results_l1 = train_evaluate_model(logistic_l1, x_train_vif, x_test_vif,␣
 ↪y_train, y_test, "Logistic With L1 (Lasso)")

# 4 Logistic Regression With L2 (Ridge)
logistic_l2 = LogisticRegression(penalty='l2', solver='liblinear', C=1.0)
```

```
results_l2 = train_evaluate_model(logistic_l2, x_train_vif, x_test_vif,␣
 ↪y_train, y_test, "Logistic With L2 (Ridge)")

# 5 Logistic Regression With Elastic Net (L1 + L2)
logistic_elasticnet = LogisticRegression(penalty='elasticnet', solver='saga',␣
 ↪l1_ratio=0.5, C=1.0,max_iter=5000)
results_elasticnet = train_evaluate_model(logistic_elasticnet, x_train_vif,␣
 ↪x_test_vif, y_train, y_test, "Logistic With Elastic Net")
# 3 Logistic Regression With L1 (Lasso) With Scaling & Vif<10
logistic_l1_1 = LogisticRegression(penalty='l1', solver='liblinear', C=1.0)
results_l1_1 = train_evaluate_model(logistic_l1_1, x_train_vif1, x_test_vif1,␣
 ↪y_train, y_test, "Logistic With L1 (Lasso) With Scaling & Vif<10")

# 4 Logistic Regression With L2 (Ridge) With Scaling & Vif<10
logistic_l2_1 = LogisticRegression(penalty='l2', solver='liblinear', C=1.0)
results_l2_1 = train_evaluate_model(logistic_l2_1, x_train_vif1, x_test_vif1,␣
 ↪y_train, y_test, "Logistic With L2 (Ridge) With Scaling & Vif<10")
# 5 Logistic Regression With Elastic Net (L1 + L2) With Scaling & Vif<10
logistic_elasticnet_1 = LogisticRegression(penalty='elasticnet', solver='saga',␣
 ↪l1_ratio=0.5, C=1.0,max_iter=5000)
results_elasticnet_1 = train_evaluate_model(logistic_elasticnet_1,␣
 ↪x_train_vif1, x_test_vif1, y_train, y_test, "Logistic With Elastic Net With␣
 ↪Scaling & Vif<10")

#

#  Step 5: PCA on Data Without VIF
# Apply PCA (retain 95% variance)
pca = PCA(n_components=0.95)
X_train_pca = x_train_df1
X_test_pca = x_test_df1

# Train Logistic Regression on PCA-transformed data
logistic_pca = LogisticRegression(solver='liblinear')
results_pca = train_evaluate_model(logistic_pca, X_train_pca, X_test_pca,␣
 ↪y_train, y_test, "Logistic With PCA")

#  Step 6: Compare Performance
results_df = pd.DataFrame([results_no_vif, results_vif, results_l1,␣
 ↪results_l2,results_l1_1,results_l2_1,results_elasticnet,results_elasticnet_1,␣
 ↪results_pca])
print("\n Model Comparison:")
print(results_df)

#  Step 7: Visualize Performance
plt.figure(figsize=(10, 5))
```

```python
# Create bar plot
ax = sns.barplot(x='Model', y='Accuracy', data=results_df, palette='viridis')

# Annotate bars with accuracy values
for p in ax.patches:
    ax.annotate(f"{p.get_height():.2f}",
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha='center', va='bottom', fontsize=12, fontweight='bold')

# Formatting
plt.xticks(rotation=90)
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy")
plt.ylim(0, 1)  # Adjust Y-axis to fit values properly
plt.show()
```

```
 Model: Logistic Without VIF
Accuracy: 0.8826666666666667
Precision: 0.8525121555915721
Recall: 0.8608837970540099
F1-score: 0.8566775244299674
AUC-ROC: 0.9163406538176181
MAE: 0.11733333333333333
MSE: 0.11733333333333333
R2_SCORE: 0.5139723737478805

Confusion Matrix:
[[798  91]
 [ 85 526]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       889
           1       0.85      0.86      0.86       611

    accuracy                           0.88      1500
   macro avg       0.88      0.88      0.88      1500
weighted avg       0.88      0.88      0.88      1500


 Model: Logistic With VIF
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
```

```
AUC-ROC: 0.9031157316464737
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L1 (Lasso)
Accuracy: 0.8513333333333334
Precision: 0.803125
Recall: 0.8412438625204582
F1-score: 0.8217426059152678
AUC-ROC: 0.9029933042330429
MAE: 0.14866666666666667
MSE: 0.14866666666666667
R2_SCORE: 0.38418090537373495

Confusion Matrix:
[[763 126]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L2 (Ridge)
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
```

```
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9031157316464737
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With Elastic Net
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9031175726602096
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L1 (Lasso) With Scaling & Vif<10
```

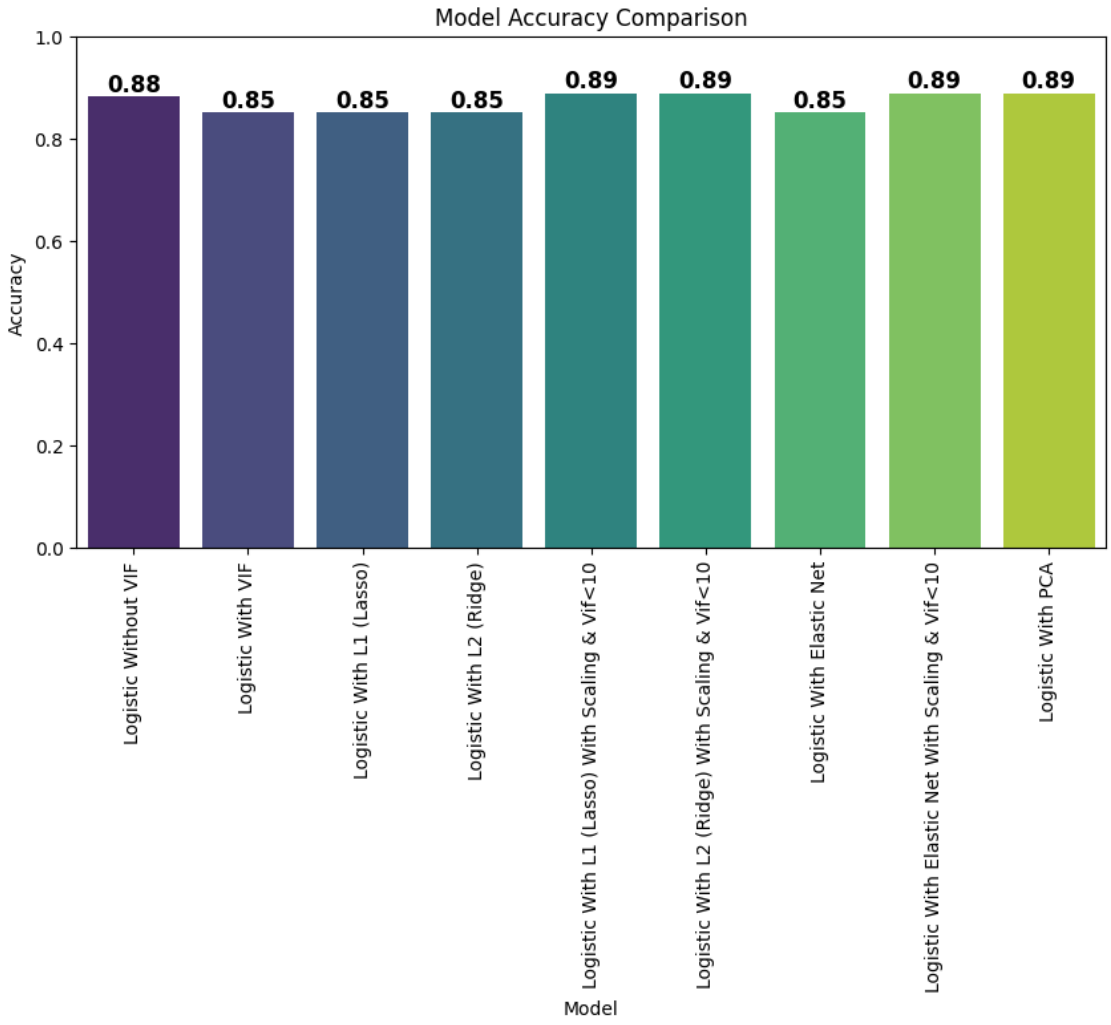
```
Accuracy: 0.888
Precision: 0.8601626016260162
Recall: 0.8657937806873978
F1-score: 0.8629690048939641
AUC-ROC: 0.9185056859709232
MAE: 0.112
MSE: 0.112
R2_SCORE: 0.5360645385775222

Confusion Matrix:
[[803  86]
 [ 82 529]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       889
           1       0.86      0.87      0.86       611

    accuracy                           0.89      1500
   macro avg       0.88      0.88      0.88      1500
weighted avg       0.89      0.89      0.89      1500


 Model: Logistic With L2 (Ridge) With Scaling & Vif<10
Accuracy: 0.888
Precision: 0.8578352180936996
Recall: 0.8690671031096563
F1-score: 0.8634146341463415
AUC-ROC: 0.9190745592152862
MAE: 0.112
MSE: 0.112
R2_SCORE: 0.5360645385775222

Confusion Matrix:
[[801  88]
 [ 80 531]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       889
           1       0.86      0.87      0.86       611

    accuracy                           0.89      1500
   macro avg       0.88      0.89      0.88      1500
weighted avg       0.89      0.89      0.89      1500
```

```
 Model: Logistic With Elastic Net With Scaling & Vif<10
Accuracy: 0.888
Precision: 0.8601626016260162
Recall: 0.8657937806873978
F1-score: 0.8629690048939641
AUC-ROC: 0.9184209993390762
MAE: 0.112
MSE: 0.112
R2_SCORE: 0.5360645385775222

Confusion Matrix:
[[803  86]
 [ 82 529]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       889
           1       0.86      0.87      0.86       611

    accuracy                           0.89      1500
   macro avg       0.88      0.88      0.88      1500
weighted avg       0.89      0.89      0.89      1500


 Model: Logistic With PCA
Accuracy: 0.888
Precision: 0.8578352180936996
Recall: 0.8690671031096563
F1-score: 0.8634146341463415
AUC-ROC: 0.9190745592152862
MAE: 0.112
MSE: 0.112
R2_SCORE: 0.5360645385775222

Confusion Matrix:
[[801  88]
 [ 80 531]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       889
           1       0.86      0.87      0.86       611

    accuracy                           0.89      1500
   macro avg       0.88      0.89      0.88      1500
```

```
weighted avg       0.89      0.89      0.89      1500
```

Model Comparison:

	Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC	MAE	MSE	R2_SCORE
0	Logistic Without VIF	0.882667	0.852512	0.860884	0.856678	0.916341	0.117333	0.117333	0.513972
1	Logistic With VIF	0.850667	0.801872	0.841244	0.821086	0.903116	0.149333	0.149333	0.381419
2	Logistic With L1 (Lasso)	0.851333	0.803125	0.841244	0.821743	0.902993	0.148667	0.148667	0.384181
3	Logistic With L2 (Ridge)	0.850667	0.801872	0.841244	0.821086	0.903116	0.149333	0.149333	0.381419
4	Logistic With L1 (Lasso) With Scaling & Vif<10	0.888000	0.860163	0.865794	0.862969	0.918506	0.112000	0.112000	0.536065
5	Logistic With L2 (Ridge) With Scaling & Vif<10	0.888000	0.857835	0.869067	0.863415	0.919075	0.112000	0.112000	0.536065
6	Logistic With Elastic Net	0.850667	0.801872	0.841244	0.821086	0.903118	0.149333	0.149333	0.381419
7	Logistic With Elastic Net With Scaling & Vif<10	0.888000	0.860163	0.865794	0.862969	0.918421	0.112000	0.112000	0.536065
8	Logistic With PCA	0.888000	0.857835	0.869067	0.863415	0.919075	0.112000	0.112000	0.536065



[44]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.linear_model import LogisticRegression

# Define the number of subplots (rows=2, columns=5)
fig, axes = plt.subplots(2, 5, figsize=(20, 10))
axes = axes.flatten()  # Flatten into a list

# Create an empty list to store results for the final table
all_results = []

for i in range(1, 10):  # Loop from _1 to _9
    x_train_var = f"x_train_vif_{i}"
    x_test_var = f"x_test_vif_{i}"
```

```
    # Train models
    logistic_vif = LogisticRegression(solver='liblinear')
    results_vif = train_evaluate_model(logistic_vif, locals()[x_train_var],␣
↪locals()[x_test_var], y_train, y_test, f"Logistic With VIF {i}")

    logistic_l1 = LogisticRegression(penalty='l1', solver='liblinear', C=1.0)
    results_l1 = train_evaluate_model(logistic_l1, locals()[x_train_var],␣
↪locals()[x_test_var], y_train, y_test, f"Logistic With L1 (Lasso) {i}")

    logistic_l2 = LogisticRegression(penalty='l2', solver='liblinear', C=1.0)
    results_l2 = train_evaluate_model(logistic_l2, locals()[x_train_var],␣
↪locals()[x_test_var], y_train, y_test, f"Logistic With L2 (Ridge) {i}")

    logistic_elasticnet = LogisticRegression(penalty='elasticnet',␣
↪solver='saga', l1_ratio=0.5, C=1.0, max_iter=5000)
    results_elasticnet = train_evaluate_model(logistic_elasticnet,␣
↪locals()[x_train_var], locals()[x_test_var], y_train, y_test, f"Logistic␣
↪With Elastic Net {i}")

    # Store results in a DataFrame
    results_df = pd.DataFrame([results_vif, results_l1, results_l2,␣
↪results_elasticnet])
    results_df['Iteration'] = i  # Add iteration number

    # Append results to the list
    all_results.append(results_df)

    #  Plot in subplot
    ax = axes[i-1]  # Use index (i-1) for correct subplot
    sns.barplot(x='Model', y='Accuracy', data=results_df, palette='viridis',␣
↪ax=ax)

    # Annotate bars with accuracy values
    for p in ax.patches:
        ax.annotate(f"{p.get_height():.2f}",
                    (p.get_x() + p.get_width() / 2, p.get_height()),
                    ha='center', va='bottom', fontsize=12, fontweight='bold')

    # Formatting
    ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
    ax.set_title(f"Model Accuracy - x_train_vif_{i}")
    ax.set_ylabel("Accuracy")
    ax.set_ylim(0, 1)

#  Remove the extra 10th (empty) subplot
```

```
fig.delaxes(axes[9])

# Adjust layout
plt.tight_layout()
plt.show()

#  Combine all results into a single table
final_results_df = pd.concat(all_results, ignore_index=True)

#  Display the final results table
print("\n Final Model Comparison Table:")
print(final_results_df)

#  Optionally, save to CSV
final_results_df.to_csv("model_comparison_results.csv", index=False)
```

```
 Model: Logistic With VIF 1
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9023692005766056
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


 Model: Logistic With L1 (Lasso) 1
Accuracy: 0.8513333333333334
Precision: 0.803125
Recall: 0.8412438625204582
F1-score: 0.8217426059152678
AUC-ROC: 0.9024824229213575
```

```
MAE: 0.14866666666666667
MSE: 0.14866666666666667
R2_SCORE: 0.38418090537373495

Confusion Matrix:
[[763 126]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L2 (Ridge) 1
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9023692005766056
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With Elastic Net 1
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
```

```
F1-score: 0.8210862619808307
AUC-ROC: 0.9023618365216622
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With VIF 2
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.8999970543780227
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L1 (Lasso) 2
Accuracy: 0.852
```

```
Precision: 0.8043818466353677
Recall: 0.8412438625204582
F1-score: 0.8223999999999999
AUC-ROC: 0.9000357156664746
MAE: 0.148
MSE: 0.148
R2_SCORE: 0.3869424259774402

Confusion Matrix:
[[764 125]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L2 (Ridge) 2
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.8999970543780227
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500
```

```
  Model: Logistic With Elastic Net 2
Accuracy: 0.8513333333333334
Precision: 0.803125
Recall: 0.8412438625204582
F1-score: 0.8217426059152678
AUC-ROC: 0.9000532052969648
MAE: 0.14866666666666667
MSE: 0.14866666666666667
R2_SCORE: 0.38418090537373495

Confusion Matrix:
[[763 126]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With VIF 3
Accuracy: 0.8553333333333333
Precision: 0.8146964856230032
Recall: 0.8346972176759411
F1-score: 0.8245755860953922
AUC-ROC: 0.8915725755229859
MAE: 0.14466666666666667
MSE: 0.14466666666666667
R2_SCORE: 0.40075002899596635

Confusion Matrix:
[[773 116]
 [101 510]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       889
           1       0.81      0.83      0.82       611

    accuracy                           0.86      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.86      0.86      0.86      1500
```

```
 Model: Logistic With L1 (Lasso) 3
Accuracy: 0.8546666666666667
Precision: 0.8144
Recall: 0.8330605564648118
F1-score: 0.8236245954692557
AUC-ROC: 0.8920052137508998
MAE: 0.14533333333333334
MSE: 0.14533333333333334
R2_SCORE: 0.3979885083922611

Confusion Matrix:
[[773 116]
 [102 509]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       889
           1       0.81      0.83      0.82       611

    accuracy                           0.85      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.86      0.85      0.85      1500


 Model: Logistic With L2 (Ridge) 3
Accuracy: 0.8553333333333333
Precision: 0.8146964856230032
Recall: 0.8346972176759411
F1-score: 0.8245755860953922
AUC-ROC: 0.8915725755229859
MAE: 0.14466666666666667
MSE: 0.14466666666666667
R2_SCORE: 0.40075002899596635

Confusion Matrix:
[[773 116]
 [101 510]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       889
           1       0.81      0.83      0.82       611

    accuracy                           0.86      1500
```

```
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.86      0.86      0.86      1500


  Model: Logistic With Elastic Net 3
Accuracy: 0.8546666666666667
Precision: 0.8133971291866029
Recall: 0.8346972176759411
F1-score: 0.8239095315024232
AUC-ROC: 0.8916756722921909
MAE: 0.14533333333333334
MSE: 0.14533333333333334
R2_SCORE: 0.3979885083922611

Confusion Matrix:
[[772 117]
 [101 510]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       889
           1       0.81      0.83      0.82       611

    accuracy                           0.85      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.86      0.85      0.85      1500


  Model: Logistic With VIF 4
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9030945599885121
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611
```

```
    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L1 (Lasso) 4
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9030282834940233
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L2 (Ridge) 4
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9030945599885121
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

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

```
              0       0.89      0.86      0.87       889
              1       0.80      0.84      0.82       611

       accuracy                           0.85      1500
      macro avg       0.84      0.85      0.85      1500
   weighted avg       0.85      0.85      0.85      1500


 Model: Logistic With Elastic Net 4
Accuracy: 0.8513333333333334
Precision: 0.8021806853582555
Recall: 0.8428805237315876
F1-score: 0.822027134876297
AUC-ROC: 0.9030595807275318
MAE: 0.14866666666666667
MSE: 0.14866666666666667
R2_SCORE: 0.38418090537373495

Confusion Matrix:
[[762 127]
 [ 96 515]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.85      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


 Model: Logistic With VIF 5
Accuracy: 0.842
Precision: 0.8006430868167203
Recall: 0.8150572831423896
F1-score: 0.8077858880778588
AUC-ROC: 0.8959173679394823
MAE: 0.158
MSE: 0.158
R2_SCORE: 0.34551961692186184

Confusion Matrix:
[[765 124]
 [113 498]]

Classification Report:
```

```
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With L1 (Lasso) 5
Accuracy: 0.842
Precision: 0.7987220447284346
Recall: 0.8183306055646481
F1-score: 0.8084074373484236
AUC-ROC: 0.8956863207156389
MAE: 0.158
MSE: 0.158
R2_SCORE: 0.34551961692186184

Confusion Matrix:
[[763 126]
 [111 500]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With L2 (Ridge) 5
Accuracy: 0.842
Precision: 0.8006430868167203
Recall: 0.8150572831423896
F1-score: 0.8077858880778588
AUC-ROC: 0.8959173679394823
MAE: 0.158
MSE: 0.158
R2_SCORE: 0.34551961692186184

Confusion Matrix:
[[765 124]
 [113 498]]
```

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

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With Elastic Net 5
Accuracy: 0.8413333333333334
Precision: 0.7984
Recall: 0.8166939443535188
F1-score: 0.8074433656957929
AUC-ROC: 0.8959302550356327
MAE: 0.15866666666666668
MSE: 0.15866666666666668
R2_SCORE: 0.34275809631815657

Confusion Matrix:
[[763 126]
 [112 499]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With VIF 6
Accuracy: 0.836
Precision: 0.7856025039123631
Recall: 0.8216039279869067
F1-score: 0.8032
AUC-ROC: 0.8848887751551514
MAE: 0.164
MSE: 0.164
R2_SCORE: 0.32066593148851485

Confusion Matrix:
```

```
[[752 137]
 [109 502]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       889
           1       0.79      0.82      0.80       611

    accuracy                           0.84      1500
   macro avg       0.83      0.83      0.83      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With L1 (Lasso) 6
Accuracy: 0.8353333333333334
Precision: 0.7861635220125787
Recall: 0.8183306055646481
F1-score: 0.8019246190858059
AUC-ROC: 0.8842904456910152
MAE: 0.16466666666666666
MSE: 0.16466666666666666
R2_SCORE: 0.3179044108848096

Confusion Matrix:
[[753 136]
 [111 500]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       889
           1       0.79      0.82      0.80       611

    accuracy                           0.84      1500
   macro avg       0.83      0.83      0.83      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With L2 (Ridge) 6
Accuracy: 0.836
Precision: 0.7856025039123631
Recall: 0.8216039279869067
F1-score: 0.8032
AUC-ROC: 0.8848887751551514
MAE: 0.164
MSE: 0.164
R2_SCORE: 0.32066593148851485
```

```
Confusion Matrix:
[[752 137]
 [109 502]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       889
           1       0.79      0.82      0.80       611

    accuracy                           0.84      1500
   macro avg       0.83      0.83      0.83      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With Elastic Net 6
Accuracy: 0.8366666666666667
Precision: 0.7886435331230284
Recall: 0.8183306055646481
F1-score: 0.8032128514056225
AUC-ROC: 0.8848722060315293
MAE: 0.16333333333333333
MSE: 0.16333333333333333
R2_SCORE: 0.32342745209222

Confusion Matrix:
[[755 134]
 [111 500]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       889
           1       0.79      0.82      0.80       611

    accuracy                           0.84      1500
   macro avg       0.83      0.83      0.83      1500
weighted avg       0.84      0.84      0.84      1500


  Model: Logistic With VIF 7
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9023498699323796
MAE: 0.14933333333333335
```

```
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L1 (Lasso) 7
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9024861049488291
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With L2 (Ridge) 7
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
```

```
AUC-ROC: 0.9023498699323796
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With Elastic Net 7
Accuracy: 0.8506666666666667
Precision: 0.8018720748829953
Recall: 0.8412438625204582
F1-score: 0.8210862619808307
AUC-ROC: 0.9023645980422661
MAE: 0.14933333333333335
MSE: 0.14933333333333335
R2_SCORE: 0.3814193847700298

Confusion Matrix:
[[762 127]
 [ 97 514]]

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.86      0.87       889
           1       0.80      0.84      0.82       611

    accuracy                           0.85      1500
   macro avg       0.84      0.85      0.85      1500
weighted avg       0.85      0.85      0.85      1500


  Model: Logistic With VIF 8
Accuracy: 0.8446666666666667
Precision: 0.8038585209003215
```

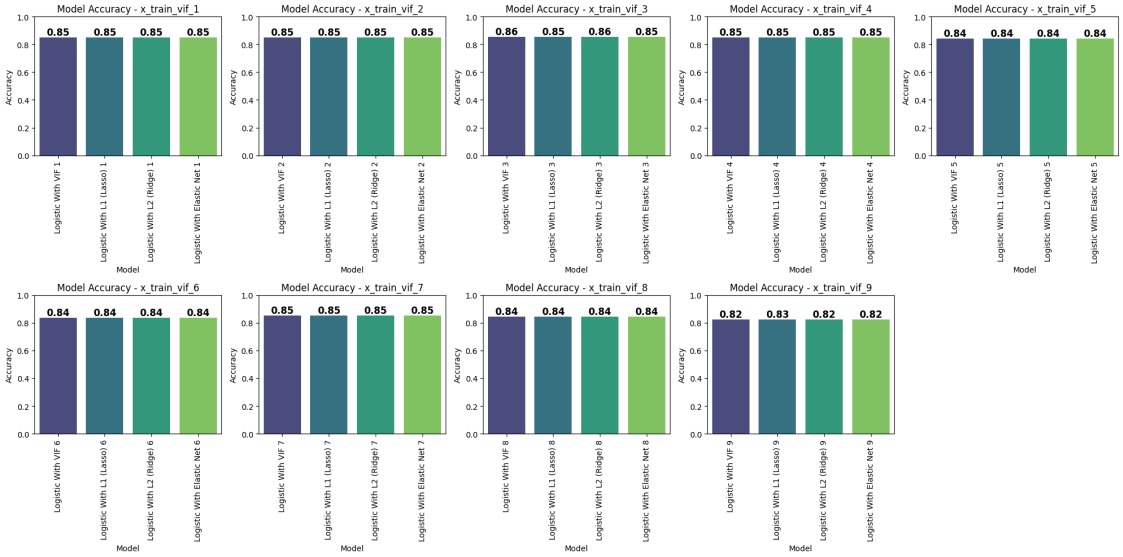
```
Recall: 0.8183306055646481
F1-score: 0.8110300081103
AUC-ROC: 0.8916535801273614
MAE: 0.15533333333333332
MSE: 0.15533333333333332
R2_SCORE: 0.3565656993366827

Confusion Matrix:
[[767 122]
 [111 500]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.85      0.84      0.84      1500


  Model: Logistic With L1 (Lasso) 8
Accuracy: 0.8446666666666667
Precision: 0.8038585209003215
Recall: 0.8183306055646481
F1-score: 0.8110300081103
AUC-ROC: 0.89168487736087
MAE: 0.15533333333333332
MSE: 0.15533333333333332
R2_SCORE: 0.3565656993366827

Confusion Matrix:
[[767 122]
 [111 500]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.85      0.84      0.84      1500


  Model: Logistic With L2 (Ridge) 8
```

```
Accuracy: 0.8446666666666667
Precision: 0.8038585209003215
Recall: 0.8183306055646481
F1-score: 0.8110300081103
AUC-ROC: 0.8916535801273614
MAE: 0.15533333333333332
MSE: 0.15533333333333332
R2_SCORE: 0.3565656993366827

Confusion Matrix:
[[767 122]
 [111 500]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.85      0.84      0.84      1500


  Model: Logistic With Elastic Net 8
Accuracy: 0.8446666666666667
Precision: 0.8038585209003215
Recall: 0.8183306055646481
F1-score: 0.8110300081103
AUC-ROC: 0.8917078900325675
MAE: 0.15533333333333332
MSE: 0.15533333333333332
R2_SCORE: 0.3565656993366827

Confusion Matrix:
[[767 122]
 [111 500]]

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       889
           1       0.80      0.82      0.81       611

    accuracy                           0.84      1500
   macro avg       0.84      0.84      0.84      1500
weighted avg       0.85      0.84      0.84      1500
```

```
 Model: Logistic With VIF 9
Accuracy: 0.822
Precision: 0.7965517241379311
Recall: 0.7561374795417348
F1-score: 0.7758186397984886
AUC-ROC: 0.8748368401576645
MAE: 0.178
MSE: 0.178
R2_SCORE: 0.2626739988107051

Confusion Matrix:
[[771 118]
 [149 462]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.87      0.85       889
           1       0.80      0.76      0.78       611

    accuracy                           0.82      1500
   macro avg       0.82      0.81      0.81      1500
weighted avg       0.82      0.82      0.82      1500


 Model: Logistic With L1 (Lasso) 9
Accuracy: 0.8253333333333334
Precision: 0.8003442340791739
Recall: 0.7610474631751227
F1-score: 0.7802013422818792
AUC-ROC: 0.874765040621968
MAE: 0.17466666666666666
MSE: 0.17466666666666666
R2_SCORE: 0.2764816018292312

Confusion Matrix:
[[773 116]
 [146 465]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.87      0.86       889
           1       0.80      0.76      0.78       611

    accuracy                           0.83      1500
   macro avg       0.82      0.82      0.82      1500
```

```
weighted avg       0.82      0.83      0.82      1500


  Model: Logistic With L2 (Ridge) 9
Accuracy: 0.822
Precision: 0.7965517241379311
Recall: 0.7561374795417348
F1-score: 0.7758186397984886
AUC-ROC: 0.8748368401576645
MAE: 0.178
MSE: 0.178
R2_SCORE: 0.2626739988107051

Confusion Matrix:
[[771 118]
 [149 462]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.87      0.85       889
           1       0.80      0.76      0.78       611

    accuracy                           0.82      1500
   macro avg       0.82      0.81      0.81      1500
weighted avg       0.82      0.82      0.82      1500


  Model: Logistic With Elastic Net 9
Accuracy: 0.8226666666666667
Precision: 0.7979274611398963
Recall: 0.7561374795417348
F1-score: 0.7764705882352941
AUC-ROC: 0.8747724046769113
MAE: 0.17733333333333334
MSE: 0.17733333333333334
R2_SCORE: 0.26543551941441035

Confusion Matrix:
[[772 117]
 [149 462]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.87      0.85       889
           1       0.80      0.76      0.78       611
```

```
    accuracy                           0.82      1500
   macro avg       0.82      0.81      0.81      1500
weighted avg       0.82      0.82      0.82      1500
```



Final Model Comparison Table:

	Model	Accuracy	Precision	Recall	F1-Score	\
0	Logistic With VIF 1	0.850667	0.801872	0.841244	0.821086	
1	Logistic With L1 (Lasso) 1	0.851333	0.803125	0.841244	0.821743	
2	Logistic With L2 (Ridge) 1	0.850667	0.801872	0.841244	0.821086	
3	Logistic With Elastic Net 1	0.850667	0.801872	0.841244	0.821086	
4	Logistic With VIF 2	0.850667	0.801872	0.841244	0.821086	
5	Logistic With L1 (Lasso) 2	0.852000	0.804382	0.841244	0.822400	
6	Logistic With L2 (Ridge) 2	0.850667	0.801872	0.841244	0.821086	
7	Logistic With Elastic Net 2	0.851333	0.803125	0.841244	0.821743	
8	Logistic With VIF 3	0.855333	0.814696	0.834697	0.824576	
9	Logistic With L1 (Lasso) 3	0.854667	0.814400	0.833061	0.823625	
10	Logistic With L2 (Ridge) 3	0.855333	0.814696	0.834697	0.824576	
11	Logistic With Elastic Net 3	0.854667	0.813397	0.834697	0.823910	
12	Logistic With VIF 4	0.850667	0.801872	0.841244	0.821086	
13	Logistic With L1 (Lasso) 4	0.850667	0.801872	0.841244	0.821086	
14	Logistic With L2 (Ridge) 4	0.850667	0.801872	0.841244	0.821086	
15	Logistic With Elastic Net 4	0.851333	0.802181	0.842881	0.822027	
16	Logistic With VIF 5	0.842000	0.800643	0.815057	0.807786	
17	Logistic With L1 (Lasso) 5	0.842000	0.798722	0.818331	0.808407	
18	Logistic With L2 (Ridge) 5	0.842000	0.800643	0.815057	0.807786	
19	Logistic With Elastic Net 5	0.841333	0.798400	0.816694	0.807443	
20	Logistic With VIF 6	0.836000	0.785603	0.821604	0.803200	

21	Logistic With L1 (Lasso)	6	0.835333	0.786164	0.818331	0.801925
22	Logistic With L2 (Ridge)	6	0.836000	0.785603	0.821604	0.803200
23	Logistic With Elastic Net	6	0.836667	0.788644	0.818331	0.803213
24	Logistic With VIF	7	0.850667	0.801872	0.841244	0.821086
25	Logistic With L1 (Lasso)	7	0.850667	0.801872	0.841244	0.821086
26	Logistic With L2 (Ridge)	7	0.850667	0.801872	0.841244	0.821086
27	Logistic With Elastic Net	7	0.850667	0.801872	0.841244	0.821086
28	Logistic With VIF	8	0.844667	0.803859	0.818331	0.811030
29	Logistic With L1 (Lasso)	8	0.844667	0.803859	0.818331	0.811030
30	Logistic With L2 (Ridge)	8	0.844667	0.803859	0.818331	0.811030
31	Logistic With Elastic Net	8	0.844667	0.803859	0.818331	0.811030
32	Logistic With VIF	9	0.822000	0.796552	0.756137	0.775819
33	Logistic With L1 (Lasso)	9	0.825333	0.800344	0.761047	0.780201
34	Logistic With L2 (Ridge)	9	0.822000	0.796552	0.756137	0.775819
35	Logistic With Elastic Net	9	0.822667	0.797927	0.756137	0.776471

	AUC-ROC	MAE	MSE	R2_SCORE	Iteration
0	0.902369	0.149333	0.149333	0.381419	1
1	0.902482	0.148667	0.148667	0.384181	1
2	0.902369	0.149333	0.149333	0.381419	1
3	0.902362	0.149333	0.149333	0.381419	1
4	0.899997	0.149333	0.149333	0.381419	2
5	0.900036	0.148000	0.148000	0.386942	2
6	0.899997	0.149333	0.149333	0.381419	2
7	0.900053	0.148667	0.148667	0.384181	2
8	0.891573	0.144667	0.144667	0.400750	3
9	0.892005	0.145333	0.145333	0.397989	3
10	0.891573	0.144667	0.144667	0.400750	3
11	0.891676	0.145333	0.145333	0.397989	3
12	0.903095	0.149333	0.149333	0.381419	4
13	0.903028	0.149333	0.149333	0.381419	4
14	0.903095	0.149333	0.149333	0.381419	4
15	0.903060	0.148667	0.148667	0.384181	4
16	0.895917	0.158000	0.158000	0.345520	5
17	0.895686	0.158000	0.158000	0.345520	5
18	0.895917	0.158000	0.158000	0.345520	5
19	0.895930	0.158667	0.158667	0.342758	5
20	0.884889	0.164000	0.164000	0.320666	6
21	0.884290	0.164667	0.164667	0.317904	6
22	0.884889	0.164000	0.164000	0.320666	6
23	0.884872	0.163333	0.163333	0.323427	6
24	0.902350	0.149333	0.149333	0.381419	7
25	0.902486	0.149333	0.149333	0.381419	7
26	0.902350	0.149333	0.149333	0.381419	7
27	0.902365	0.149333	0.149333	0.381419	7
28	0.891654	0.155333	0.155333	0.356566	8
29	0.891685	0.155333	0.155333	0.356566	8
30	0.891654	0.155333	0.155333	0.356566	8

```
31  0.891708  0.155333  0.155333  0.356566          8
32  0.874837  0.178000  0.178000  0.262674          9
33  0.874765  0.174667  0.174667  0.276482          9
34  0.874837  0.178000  0.178000  0.262674          9
35  0.874772  0.177333  0.177333  0.265436          9
```

[45]:
```
#There is no significant change in model performance with dropping the highly␣
 ↪correlated factors.
```

[46]:
```
#DECISION TREE:
model = DecisionTreeClassifier(max_depth=5)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
dt = DecisionTreeClassifier(criterion='gini', max_depth=5, min_samples_split=10)
dt.fit(x_train, y_train)
y_pred1 = dt.predict(x_test)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
accuracy1 = accuracy_score(y_test, y_pred1)
print(f"Decision Tree Accuracy: {accuracy:.3f}")
print(f"Decision Tree Accuracy: {accuracy1:.3f}")
```

```
Decision Tree Accuracy: 0.864
Decision Tree Accuracy: 0.865
```

[47]:
```
# RANDOM FOREST:
model = RandomForestClassifier(n_estimators=100)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"Decision Tree Accuracy: {accuracy:.3f}")
```

```
Decision Tree Accuracy: 0.901
```

[48]:
```
#XGBOOST CLASSIFICATION:
from xgboost import XGBClassifier
model = XGBClassifier(scale_pos_weight=10)  # Adjust for class imbalance
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"XG BOOST CLASS: {accuracy:.3f}")
```

```
XG BOOST CLASS: 0.878
```

[49]:
```
#LGM CLASSIFIER:
import lightgbm as lgb
from lightgbm import LGBMClassifier
```

```
model = LGBMClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"XG BOOST CLASS: {accuracy:.3f}")
```

```
[LightGBM] [Info] Number of positive: 1426, number of negative: 2074
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.004211 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 594
[LightGBM] [Info] Number of data points in the train set: 3500, number of used
features: 17
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.407429 -> initscore=-0.374606
[LightGBM] [Info] Start training from score -0.374606
XG BOOST CLASS: 0.895
```

[50]:
```
from catboost import CatBoostClassifier
model = CatBoostClassifier(verbose=0)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"CAT BOOST CLASS: {accuracy:.3f}")
```

```
CAT BOOST CLASS: 0.900
```

[51]:
```
from sklearn.svm import SVC

model = SVC(kernel='rbf', probability=True, class_weight='balanced')

model.fit(x_train_df1, y_train)
y_pred = model.predict(x_test_df1)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"SVC CLASS: {accuracy:.3f}")
auc = roc_auc_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred.round())


print(auc)
print(f1)
```

```
SVC CLASS: 0.878
0.8783945992021047
0.8546465448768865
```

[52]:
```
print(np.unique(y_pred, return_counts=True))  # Check class distribution
```

```
(array([0, 1]), array([852, 648]))
```

[53]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=10)
model.fit(x_train_df1, y_train)
y_pred = model.predict(x_test_df1)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"KNN CLASS: {accuracy:.3f}")
```

```
KNN CLASS: 0.857
```

[54]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense ,Input

# Function to create and compile a Sequential model
def get_seq_dense():
    model = Sequential([
         Input(shape=(x_train.shape[1],)),  # Explicit Input layer
        Dense(32, activation='relu'),
        Dense(16, activation='relu'),
        Dense(1, activation='sigmoid')   # Binary classification
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model  # Correct indentation

# Create model instance
model = get_seq_dense()  # Call the function to get a fresh compiled model

# Train the model
model.fit(x_train, y_train, epochs=50, batch_size=32)

# Predict
y_pred = model.predict(x_test)

# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Seq Bin Adam: {accuracy:.4f}")
```

```
Epoch 1/50
110/110             1s 2ms/step -
accuracy: 0.4859 - loss: 4.4266
Epoch 2/50
110/110             0s 2ms/step -
accuracy: 0.6306 - loss: 0.6497
Epoch 3/50
```

```
110/110             0s 2ms/step -
accuracy: 0.6557 - loss: 0.6254
Epoch 4/50
110/110             0s 2ms/step -
accuracy: 0.6733 - loss: 0.6037
Epoch 5/50
110/110             0s 2ms/step -
accuracy: 0.7184 - loss: 0.5760
Epoch 6/50
110/110             0s 2ms/step -
accuracy: 0.7117 - loss: 0.5655
Epoch 7/50
110/110             0s 2ms/step -
accuracy: 0.7589 - loss: 0.5255
Epoch 8/50
110/110             0s 1ms/step -
accuracy: 0.7749 - loss: 0.4990
Epoch 9/50
110/110             0s 2ms/step -
accuracy: 0.8023 - loss: 0.4703
Epoch 10/50
110/110             0s 2ms/step -
accuracy: 0.7966 - loss: 0.4676
Epoch 11/50
110/110             0s 2ms/step -
accuracy: 0.8209 - loss: 0.4426
Epoch 12/50
110/110             0s 2ms/step -
accuracy: 0.8336 - loss: 0.4273
Epoch 13/50
110/110             0s 2ms/step -
accuracy: 0.8365 - loss: 0.4165
Epoch 14/50
110/110             0s 2ms/step -
accuracy: 0.8429 - loss: 0.4070
Epoch 15/50
110/110             0s 2ms/step -
accuracy: 0.8630 - loss: 0.3919
Epoch 16/50
110/110             0s 2ms/step -
accuracy: 0.8748 - loss: 0.3880
Epoch 17/50
110/110             0s 2ms/step -
accuracy: 0.8794 - loss: 0.3629
Epoch 18/50
110/110             0s 2ms/step -
accuracy: 0.8675 - loss: 0.3882
Epoch 19/50
```

```
110/110             0s 2ms/step -
accuracy: 0.8607 - loss: 0.3900
Epoch 20/50
110/110             0s 2ms/step -
accuracy: 0.8773 - loss: 0.3860
Epoch 21/50
110/110             0s 2ms/step -
accuracy: 0.8700 - loss: 0.3802
Epoch 22/50
110/110             0s 2ms/step -
accuracy: 0.8758 - loss: 0.3773
Epoch 23/50
110/110             0s 2ms/step -
accuracy: 0.8664 - loss: 0.3863
Epoch 24/50
110/110             0s 2ms/step -
accuracy: 0.8748 - loss: 0.3773
Epoch 25/50
110/110             0s 2ms/step -
accuracy: 0.8733 - loss: 0.3701
Epoch 26/50
110/110             0s 2ms/step -
accuracy: 0.8936 - loss: 0.3639
Epoch 27/50
110/110             0s 2ms/step -
accuracy: 0.8817 - loss: 0.3766
Epoch 28/50
110/110             0s 2ms/step -
accuracy: 0.8562 - loss: 0.3917
Epoch 29/50
110/110             0s 2ms/step -
accuracy: 0.8716 - loss: 0.3931
Epoch 30/50
110/110             0s 2ms/step -
accuracy: 0.8870 - loss: 0.3583
Epoch 31/50
110/110             0s 2ms/step -
accuracy: 0.8807 - loss: 0.3612
Epoch 32/50
110/110             0s 2ms/step -
accuracy: 0.8896 - loss: 0.3616
Epoch 33/50
110/110             0s 2ms/step -
accuracy: 0.8839 - loss: 0.3520
Epoch 34/50
110/110             0s 2ms/step -
accuracy: 0.8908 - loss: 0.3575
Epoch 35/50
```

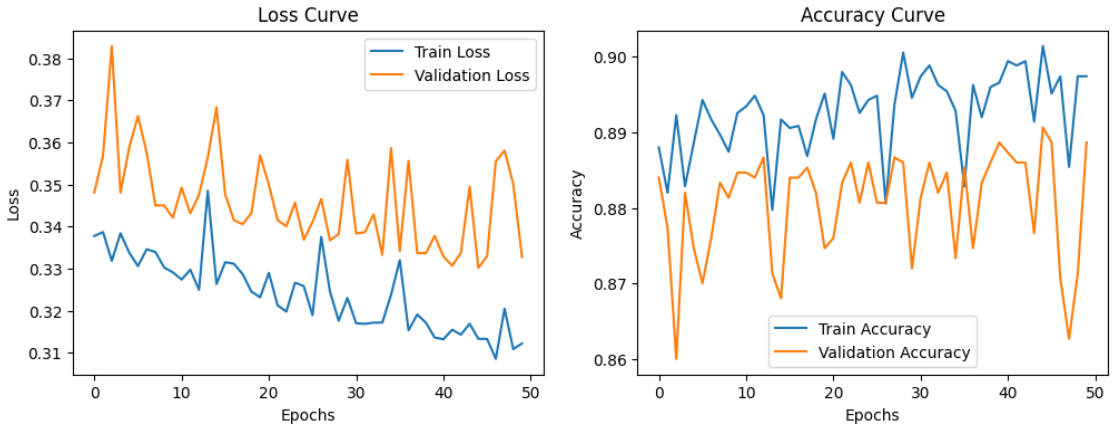
```
110/110             0s 2ms/step -
accuracy: 0.8833 - loss: 0.3486
Epoch 36/50
110/110             0s 2ms/step -
accuracy: 0.8823 - loss: 0.3565
Epoch 37/50
110/110             0s 2ms/step -
accuracy: 0.8751 - loss: 0.3377
Epoch 38/50
110/110             0s 2ms/step -
accuracy: 0.8924 - loss: 0.3279
Epoch 39/50
110/110             0s 2ms/step -
accuracy: 0.8859 - loss: 0.3445
Epoch 40/50
110/110             0s 2ms/step -
accuracy: 0.8960 - loss: 0.3383
Epoch 41/50
110/110             0s 2ms/step -
accuracy: 0.8857 - loss: 0.3600
Epoch 42/50
110/110             0s 2ms/step -
accuracy: 0.8942 - loss: 0.3320
Epoch 43/50
110/110             0s 2ms/step -
accuracy: 0.8920 - loss: 0.3290
Epoch 44/50
110/110             0s 2ms/step -
accuracy: 0.8976 - loss: 0.3268
Epoch 45/50
110/110             0s 2ms/step -
accuracy: 0.8912 - loss: 0.3334
Epoch 46/50
110/110             0s 2ms/step -
accuracy: 0.8970 - loss: 0.3288
Epoch 47/50
110/110             0s 2ms/step -
accuracy: 0.8953 - loss: 0.3275
Epoch 48/50
110/110             0s 2ms/step -
accuracy: 0.8784 - loss: 0.3421
Epoch 49/50
110/110             0s 1ms/step -
accuracy: 0.8959 - loss: 0.3225
Epoch 50/50
110/110             0s 2ms/step -
accuracy: 0.8935 - loss: 0.3390
47/47             0s 2ms/step
```

```
47/47              0s 2ms/step -
accuracy: 0.8873 - loss: 0.3361
Seq Bin Adam: 0.8873
```

[55]:
```
y_pred_prob = model.predict(x_test)  # Probabilities
y_pred = (y_pred_prob > 0.5).astype(int)  # Convert to binary (0 or 1)

# Print first 10 predictions
print("Predicted Probabilities:", y_pred_prob[:10].flatten())
print("Predicted Labels:", y_pred[:10].flatten())
```

```
47/47              0s 2ms/step
Predicted Probabilities: [0.20803893 0.06927779 0.4409396  0.94762105 0.84928834
0.02158139
 0.1264135  0.01916538 0.8181076  0.0217882 ]
Predicted Labels: [0 0 0 1 1 0 0 0 1 0]
```

[56]:
```
history = model.fit(x_train, y_train, epochs=50, batch_size=32,␣
 ↪validation_data=(x_test, y_test), verbose=0)

plt.figure(figsize=(12, 4))

# Loss Curve
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Curve')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Accuracy Curve
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Curve')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



[57]:
```
from tensorflow.keras.layers import Dropout
# Function to create a Sequential Dropout model
def get_seq_dropout():
    model = Sequential([
        Input(shape=(x_train.shape[1],)),  #  Explicit input layer
        Dense(64, activation='relu'),
        Dropout(0.3),
        Dense(32, activation='relu'),
        Dropout(0.3),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model


# Create model instance
model = get_seq_dropout()  # Call the function to get a fresh compiled model

# Train the model
model.fit(x_train, y_train, epochs=50, batch_size=32)

# Predict
y_pred = model.predict(x_test)

# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Seq Bin Adam: {accuracy:.4f}")
```

```
Epoch 1/50
110/110             1s 2ms/step -
accuracy: 0.5160 - loss: 2.4779
```

```
Epoch 2/50
110/110             0s 2ms/step -
accuracy: 0.5382 - loss: 0.8176
Epoch 3/50
110/110             0s 2ms/step -
accuracy: 0.5580 - loss: 0.7092
Epoch 4/50
110/110             0s 2ms/step -
accuracy: 0.5809 - loss: 0.6895
Epoch 5/50
110/110             0s 2ms/step -
accuracy: 0.5843 - loss: 0.6821
Epoch 6/50
110/110             0s 2ms/step -
accuracy: 0.5899 - loss: 0.6791
Epoch 7/50
110/110             0s 2ms/step -
accuracy: 0.5757 - loss: 0.6796
Epoch 8/50
110/110             0s 2ms/step -
accuracy: 0.5866 - loss: 0.6801
Epoch 9/50
110/110             0s 2ms/step -
accuracy: 0.5960 - loss: 0.6780
Epoch 10/50
110/110             0s 2ms/step -
accuracy: 0.5779 - loss: 0.6749
Epoch 11/50
110/110             0s 2ms/step -
accuracy: 0.6054 - loss: 0.6612
Epoch 12/50
110/110             0s 2ms/step -
accuracy: 0.6198 - loss: 0.6481
Epoch 13/50
110/110             0s 2ms/step -
accuracy: 0.6276 - loss: 0.6390
Epoch 14/50
110/110             0s 2ms/step -
accuracy: 0.6286 - loss: 0.6386
Epoch 15/50
110/110             0s 2ms/step -
accuracy: 0.6450 - loss: 0.6197
Epoch 16/50
110/110             0s 2ms/step -
accuracy: 0.6870 - loss: 0.5925
Epoch 17/50
110/110             0s 2ms/step -
accuracy: 0.6819 - loss: 0.5743
```

```
Epoch 18/50
110/110             0s 2ms/step -
accuracy: 0.7328 - loss: 0.5249
Epoch 19/50
110/110             0s 2ms/step -
accuracy: 0.7873 - loss: 0.4886
Epoch 20/50
110/110             0s 2ms/step -
accuracy: 0.8069 - loss: 0.4637
Epoch 21/50
110/110             0s 2ms/step -
accuracy: 0.8275 - loss: 0.4267
Epoch 22/50
110/110             0s 2ms/step -
accuracy: 0.8103 - loss: 0.4523
Epoch 23/50
110/110             0s 2ms/step -
accuracy: 0.8366 - loss: 0.4107
Epoch 24/50
110/110             0s 2ms/step -
accuracy: 0.8451 - loss: 0.4150
Epoch 25/50
110/110             0s 2ms/step -
accuracy: 0.8582 - loss: 0.3955
Epoch 26/50
110/110             0s 2ms/step -
accuracy: 0.8492 - loss: 0.4113
Epoch 27/50
110/110             0s 2ms/step -
accuracy: 0.8608 - loss: 0.3951
Epoch 28/50
110/110             0s 2ms/step -
accuracy: 0.8749 - loss: 0.3605
Epoch 29/50
110/110             0s 2ms/step -
accuracy: 0.8719 - loss: 0.3719
Epoch 30/50
110/110             0s 2ms/step -
accuracy: 0.8756 - loss: 0.3854
Epoch 31/50
110/110             0s 2ms/step -
accuracy: 0.8538 - loss: 0.3813
Epoch 32/50
110/110             0s 2ms/step -
accuracy: 0.8752 - loss: 0.3723
Epoch 33/50
110/110             0s 2ms/step -
accuracy: 0.8752 - loss: 0.3827
```

```
Epoch 34/50
110/110             0s 2ms/step -
accuracy: 0.8796 - loss: 0.3551
Epoch 35/50
110/110             0s 2ms/step -
accuracy: 0.8724 - loss: 0.3694
Epoch 36/50
110/110             0s 2ms/step -
accuracy: 0.8705 - loss: 0.3739
Epoch 37/50
110/110             0s 2ms/step -
accuracy: 0.8795 - loss: 0.3431
Epoch 38/50
110/110             0s 2ms/step -
accuracy: 0.8879 - loss: 0.3528
Epoch 39/50
110/110             0s 2ms/step -
accuracy: 0.8666 - loss: 0.3761
Epoch 40/50
110/110             0s 2ms/step -
accuracy: 0.8835 - loss: 0.3577
Epoch 41/50
110/110             0s 2ms/step -
accuracy: 0.8758 - loss: 0.3626
Epoch 42/50
110/110             0s 2ms/step -
accuracy: 0.8808 - loss: 0.3425
Epoch 43/50
110/110             0s 2ms/step -
accuracy: 0.8865 - loss: 0.3399
Epoch 44/50
110/110             0s 2ms/step -
accuracy: 0.8843 - loss: 0.3465
Epoch 45/50
110/110             0s 2ms/step -
accuracy: 0.8799 - loss: 0.3513
Epoch 46/50
110/110             0s 2ms/step -
accuracy: 0.8892 - loss: 0.3468
Epoch 47/50
110/110             0s 2ms/step -
accuracy: 0.8790 - loss: 0.3675
Epoch 48/50
110/110             0s 2ms/step -
accuracy: 0.8770 - loss: 0.3614
Epoch 49/50
110/110             0s 2ms/step -
accuracy: 0.8754 - loss: 0.3593
```

```
Epoch 50/50
110/110             0s 2ms/step -
accuracy: 0.8837 - loss: 0.3435
47/47             0s 2ms/step
47/47             0s 1ms/step -
accuracy: 0.8950 - loss: 0.3257
Seq Bin Adam: 0.8873
```

[58]:
```
#NAIVE BAYES :
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"NAIVE BAYES: {accuracy:.3f}")
```

```
NAIVE BAYES: 0.853
```

[59]:
```
models = {
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "XGBoost": XGBClassifier(),
    "LGB": LGBMClassifier(),
    "CAT": CatBoostClassifier(),
    "SVM": SVC(probability=True),
    "KNN": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
    "SEQ DENSE": get_seq_dense(),
    "SEQ DROPOUT": get_seq_dropout()

}




for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)

    # Check if the model has predict_proba()
    if hasattr(model, "predict_proba"):
        y_pred_prob = model.predict_proba(x_test)[:, 1]
    else:
        y_pred_prob = model.predict(x_test).flatten()  # Use raw probabilities␣
 ↪for NN
```

```
    auc = roc_auc_score(y_test, y_pred_prob)
    f1 = f1_score(y_test, y_pred.round())  # Round predictions for F1 score

    print(f"{name}: AUC={auc:.3f}, F1={f1:.3f}")
```

```
Decision Tree: AUC=0.813, F1=0.779
Random Forest: AUC=0.918, F1=0.877
XGBoost: AUC=0.909, F1=0.863
[LightGBM] [Info] Number of positive: 1426, number of negative: 2074
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000539 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 594
[LightGBM] [Info] Number of data points in the train set: 3500, number of used
features: 17
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.407429 -> initscore=-0.374606
[LightGBM] [Info] Start training from score -0.374606
LGB: AUC=0.910, F1=0.871
Learning rate set to 0.017589
0:      learn: 0.6788273        total: 2.63ms   remaining: 2.63s
1:      learn: 0.6649787        total: 5.28ms   remaining: 2.63s
2:      learn: 0.6493541        total: 7.83ms   remaining: 2.6s
3:      learn: 0.6344958        total: 10.5ms   remaining: 2.62s
4:      learn: 0.6202384        total: 13ms     remaining: 2.59s
5:      learn: 0.6066893        total: 15.4ms   remaining: 2.56s
6:      learn: 0.5946300        total: 18.6ms   remaining: 2.64s
7:      learn: 0.5833650        total: 21.1ms   remaining: 2.62s
8:      learn: 0.5727236        total: 23.8ms   remaining: 2.62s
9:      learn: 0.5612988        total: 26.1ms   remaining: 2.59s
10:     learn: 0.5504562        total: 28.7ms   remaining: 2.58s
11:     learn: 0.5402509        total: 31.3ms   remaining: 2.58s
12:     learn: 0.5326323        total: 33.8ms   remaining: 2.57s
13:     learn: 0.5247246        total: 37ms     remaining: 2.61s
14:     learn: 0.5156423        total: 40.3ms   remaining: 2.64s
15:     learn: 0.5069254        total: 42.7ms   remaining: 2.63s
16:     learn: 0.5018860        total: 45.2ms   remaining: 2.62s
17:     learn: 0.4939638        total: 47.6ms   remaining: 2.6s
18:     learn: 0.4867092        total: 50.3ms   remaining: 2.6s
19:     learn: 0.4799832        total: 52.8ms   remaining: 2.58s
20:     learn: 0.4737642        total: 55.7ms   remaining: 2.6s
21:     learn: 0.4669738        total: 58.2ms   remaining: 2.58s
22:     learn: 0.4604536        total: 60.6ms   remaining: 2.58s
23:     learn: 0.4547818        total: 63ms     remaining: 2.56s
24:     learn: 0.4486030        total: 65.6ms   remaining: 2.56s
25:     learn: 0.4435899        total: 68.2ms   remaining: 2.55s
```

```
26:     learn: 0.4391185        total: 70.7ms   remaining: 2.55s
27:     learn: 0.4346487        total: 73.3ms   remaining: 2.54s
28:     learn: 0.4293706        total: 75.9ms   remaining: 2.54s
29:     learn: 0.4259852        total: 78ms     remaining: 2.52s
30:     learn: 0.4212334        total: 80.7ms   remaining: 2.52s
31:     learn: 0.4171080        total: 83ms     remaining: 2.51s
32:     learn: 0.4133286        total: 85.4ms   remaining: 2.5s
33:     learn: 0.4099061        total: 88.2ms   remaining: 2.5s
34:     learn: 0.4058518        total: 90.7ms   remaining: 2.5s
35:     learn: 0.4022870        total: 93.1ms   remaining: 2.49s
36:     learn: 0.3989420        total: 95.6ms   remaining: 2.49s
37:     learn: 0.3950763        total: 98.5ms   remaining: 2.49s
38:     learn: 0.3914197        total: 101ms    remaining: 2.49s
39:     learn: 0.3878956        total: 104ms    remaining: 2.49s
40:     learn: 0.3853162        total: 106ms    remaining: 2.48s
41:     learn: 0.3818756        total: 109ms    remaining: 2.49s
42:     learn: 0.3786759        total: 112ms    remaining: 2.49s
43:     learn: 0.3755497        total: 114ms    remaining: 2.48s
44:     learn: 0.3725590        total: 117ms    remaining: 2.48s
45:     learn: 0.3697987        total: 119ms    remaining: 2.48s
46:     learn: 0.3671974        total: 122ms    remaining: 2.47s
47:     learn: 0.3645659        total: 124ms    remaining: 2.47s
48:     learn: 0.3619649        total: 127ms    remaining: 2.46s
49:     learn: 0.3594704        total: 129ms    remaining: 2.46s
50:     learn: 0.3570575        total: 132ms    remaining: 2.46s
51:     learn: 0.3553168        total: 135ms    remaining: 2.47s
52:     learn: 0.3533582        total: 138ms    remaining: 2.46s
53:     learn: 0.3511774        total: 140ms    remaining: 2.46s
54:     learn: 0.3489681        total: 143ms    remaining: 2.45s
55:     learn: 0.3470001        total: 145ms    remaining: 2.45s
56:     learn: 0.3448833        total: 148ms    remaining: 2.45s
57:     learn: 0.3430067        total: 151ms    remaining: 2.44s
58:     learn: 0.3410832        total: 153ms    remaining: 2.44s
59:     learn: 0.3398477        total: 156ms    remaining: 2.44s
60:     learn: 0.3380265        total: 158ms    remaining: 2.43s
61:     learn: 0.3369886        total: 161ms    remaining: 2.43s
62:     learn: 0.3352829        total: 163ms    remaining: 2.42s
63:     learn: 0.3340597        total: 165ms    remaining: 2.42s
64:     learn: 0.3328972        total: 168ms    remaining: 2.41s
65:     learn: 0.3312876        total: 170ms    remaining: 2.4s
66:     learn: 0.3298567        total: 172ms    remaining: 2.4s
67:     learn: 0.3288077        total: 175ms    remaining: 2.4s
68:     learn: 0.3279073        total: 179ms    remaining: 2.41s
69:     learn: 0.3268354        total: 181ms    remaining: 2.41s
70:     learn: 0.3252381        total: 184ms    remaining: 2.41s
71:     learn: 0.3241861        total: 187ms    remaining: 2.41s
72:     learn: 0.3228455        total: 189ms    remaining: 2.4s
73:     learn: 0.3218462        total: 192ms    remaining: 2.4s
```

```
74:     learn: 0.3204267        total: 195ms    remaining: 2.4s
75:     learn: 0.3194255        total: 197ms    remaining: 2.39s
76:     learn: 0.3180770        total: 199ms    remaining: 2.39s
77:     learn: 0.3174500        total: 202ms    remaining: 2.39s
78:     learn: 0.3164306        total: 205ms    remaining: 2.39s
79:     learn: 0.3156650        total: 208ms    remaining: 2.39s
80:     learn: 0.3148097        total: 211ms    remaining: 2.39s
81:     learn: 0.3141221        total: 213ms    remaining: 2.39s
82:     learn: 0.3136010        total: 216ms    remaining: 2.38s
83:     learn: 0.3124615        total: 218ms    remaining: 2.38s
84:     learn: 0.3116281        total: 221ms    remaining: 2.38s
85:     learn: 0.3106201        total: 224ms    remaining: 2.38s
86:     learn: 0.3094467        total: 226ms    remaining: 2.37s
87:     learn: 0.3084743        total: 229ms    remaining: 2.37s
88:     learn: 0.3077030        total: 232ms    remaining: 2.37s
89:     learn: 0.3069185        total: 234ms    remaining: 2.37s
90:     learn: 0.3063879        total: 237ms    remaining: 2.37s
91:     learn: 0.3059865        total: 240ms    remaining: 2.37s
92:     learn: 0.3051843        total: 242ms    remaining: 2.36s
93:     learn: 0.3043486        total: 245ms    remaining: 2.36s
94:     learn: 0.3040257        total: 247ms    remaining: 2.35s
95:     learn: 0.3031204        total: 250ms    remaining: 2.35s
96:     learn: 0.3026046        total: 252ms    remaining: 2.35s
97:     learn: 0.3017879        total: 255ms    remaining: 2.35s
98:     learn: 0.3012167        total: 258ms    remaining: 2.34s
99:     learn: 0.3008170        total: 260ms    remaining: 2.34s
100:    learn: 0.3002719        total: 263ms    remaining: 2.34s
101:    learn: 0.2997426        total: 265ms    remaining: 2.34s
102:    learn: 0.2992654        total: 268ms    remaining: 2.33s
103:    learn: 0.2984848        total: 270ms    remaining: 2.33s
104:    learn: 0.2977971        total: 273ms    remaining: 2.33s
105:    learn: 0.2972975        total: 275ms    remaining: 2.32s
106:    learn: 0.2969422        total: 278ms    remaining: 2.32s
107:    learn: 0.2964837        total: 280ms    remaining: 2.31s
108:    learn: 0.2955981        total: 282ms    remaining: 2.31s
109:    learn: 0.2949189        total: 285ms    remaining: 2.3s
110:    learn: 0.2943224        total: 287ms    remaining: 2.3s
111:    learn: 0.2937766        total: 289ms    remaining: 2.29s
112:    learn: 0.2935197        total: 292ms    remaining: 2.29s
113:    learn: 0.2931671        total: 294ms    remaining: 2.29s
114:    learn: 0.2925807        total: 297ms    remaining: 2.28s
115:    learn: 0.2919801        total: 299ms    remaining: 2.28s
116:    learn: 0.2917302        total: 301ms    remaining: 2.27s
117:    learn: 0.2913559        total: 304ms    remaining: 2.27s
118:    learn: 0.2910424        total: 306ms    remaining: 2.27s
119:    learn: 0.2908663        total: 309ms    remaining: 2.26s
120:    learn: 0.2905349        total: 311ms    remaining: 2.26s
121:    learn: 0.2901225        total: 313ms    remaining: 2.25s
```

```
122:    learn: 0.2897949        total: 316ms    remaining: 2.25s
123:    learn: 0.2894964        total: 318ms    remaining: 2.25s
124:    learn: 0.2890526        total: 321ms    remaining: 2.25s
125:    learn: 0.2885858        total: 323ms    remaining: 2.24s
126:    learn: 0.2882697        total: 326ms    remaining: 2.24s
127:    learn: 0.2877257        total: 328ms    remaining: 2.24s
128:    learn: 0.2874386        total: 331ms    remaining: 2.24s
129:    learn: 0.2868708        total: 334ms    remaining: 2.23s
130:    learn: 0.2863531        total: 337ms    remaining: 2.23s
131:    learn: 0.2859087        total: 340ms    remaining: 2.23s
132:    learn: 0.2855177        total: 342ms    remaining: 2.23s
133:    learn: 0.2851770        total: 344ms    remaining: 2.22s
134:    learn: 0.2846793        total: 347ms    remaining: 2.22s
135:    learn: 0.2839572        total: 349ms    remaining: 2.22s
136:    learn: 0.2836962        total: 352ms    remaining: 2.21s
137:    learn: 0.2831926        total: 354ms    remaining: 2.21s
138:    learn: 0.2828400        total: 357ms    remaining: 2.21s
139:    learn: 0.2826417        total: 360ms    remaining: 2.21s
140:    learn: 0.2823871        total: 363ms    remaining: 2.21s
141:    learn: 0.2820277        total: 367ms    remaining: 2.21s
142:    learn: 0.2816384        total: 369ms    remaining: 2.21s
143:    learn: 0.2813431        total: 372ms    remaining: 2.21s
144:    learn: 0.2810055        total: 375ms    remaining: 2.21s
145:    learn: 0.2807760        total: 377ms    remaining: 2.21s
146:    learn: 0.2804262        total: 379ms    remaining: 2.2s
147:    learn: 0.2801527        total: 382ms    remaining: 2.2s
148:    learn: 0.2798307        total: 385ms    remaining: 2.2s
149:    learn: 0.2794412        total: 387ms    remaining: 2.19s
150:    learn: 0.2793273        total: 389ms    remaining: 2.19s
151:    learn: 0.2790201        total: 392ms    remaining: 2.19s
152:    learn: 0.2786454        total: 394ms    remaining: 2.18s
153:    learn: 0.2784560        total: 397ms    remaining: 2.18s
154:    learn: 0.2782214        total: 399ms    remaining: 2.18s
155:    learn: 0.2779689        total: 402ms    remaining: 2.17s
156:    learn: 0.2776415        total: 404ms    remaining: 2.17s
157:    learn: 0.2773798        total: 407ms    remaining: 2.17s
158:    learn: 0.2768942        total: 409ms    remaining: 2.16s
159:    learn: 0.2766475        total: 411ms    remaining: 2.16s
160:    learn: 0.2764649        total: 414ms    remaining: 2.16s
161:    learn: 0.2760938        total: 416ms    remaining: 2.15s
162:    learn: 0.2759384        total: 419ms    remaining: 2.15s
163:    learn: 0.2758069        total: 422ms    remaining: 2.15s
164:    learn: 0.2755624        total: 425ms    remaining: 2.15s
165:    learn: 0.2752991        total: 428ms    remaining: 2.15s
166:    learn: 0.2751204        total: 430ms    remaining: 2.15s
167:    learn: 0.2748173        total: 433ms    remaining: 2.14s
168:    learn: 0.2744734        total: 436ms    remaining: 2.14s
169:    learn: 0.2741649        total: 438ms    remaining: 2.14s
```

```
170:    learn: 0.2738601        total: 441ms    remaining: 2.14s
171:    learn: 0.2735586        total: 444ms    remaining: 2.13s
172:    learn: 0.2733720        total: 446ms    remaining: 2.13s
173:    learn: 0.2731516        total: 449ms    remaining: 2.13s
174:    learn: 0.2729082        total: 452ms    remaining: 2.13s
175:    learn: 0.2726985        total: 454ms    remaining: 2.13s
176:    learn: 0.2725077        total: 457ms    remaining: 2.13s
177:    learn: 0.2721814        total: 459ms    remaining: 2.12s
178:    learn: 0.2719885        total: 462ms    remaining: 2.12s
179:    learn: 0.2718128        total: 465ms    remaining: 2.12s
180:    learn: 0.2716985        total: 467ms    remaining: 2.11s
181:    learn: 0.2713498        total: 470ms    remaining: 2.11s
182:    learn: 0.2709675        total: 473ms    remaining: 2.11s
183:    learn: 0.2706791        total: 475ms    remaining: 2.11s
184:    learn: 0.2704180        total: 478ms    remaining: 2.11s
185:    learn: 0.2703013        total: 481ms    remaining: 2.1s
186:    learn: 0.2700513        total: 484ms    remaining: 2.1s
187:    learn: 0.2699126        total: 486ms    remaining: 2.1s
188:    learn: 0.2697431        total: 489ms    remaining: 2.1s
189:    learn: 0.2695194        total: 492ms    remaining: 2.1s
190:    learn: 0.2693236        total: 494ms    remaining: 2.09s
191:    learn: 0.2690501        total: 497ms    remaining: 2.09s
192:    learn: 0.2688170        total: 500ms    remaining: 2.09s
193:    learn: 0.2686461        total: 503ms    remaining: 2.09s
194:    learn: 0.2685019        total: 505ms    remaining: 2.09s
195:    learn: 0.2683251        total: 508ms    remaining: 2.08s
196:    learn: 0.2680712        total: 510ms    remaining: 2.08s
197:    learn: 0.2678056        total: 512ms    remaining: 2.08s
198:    learn: 0.2675346        total: 515ms    remaining: 2.07s
199:    learn: 0.2673429        total: 517ms    remaining: 2.07s
200:    learn: 0.2668981        total: 520ms    remaining: 2.06s
201:    learn: 0.2666950        total: 523ms    remaining: 2.06s
202:    learn: 0.2665297        total: 525ms    remaining: 2.06s
203:    learn: 0.2663308        total: 528ms    remaining: 2.06s
204:    learn: 0.2661097        total: 530ms    remaining: 2.06s
205:    learn: 0.2658672        total: 533ms    remaining: 2.05s
206:    learn: 0.2655339        total: 535ms    remaining: 2.05s
207:    learn: 0.2652896        total: 538ms    remaining: 2.05s
208:    learn: 0.2651590        total: 540ms    remaining: 2.04s
209:    learn: 0.2650090        total: 542ms    remaining: 2.04s
210:    learn: 0.2646476        total: 545ms    remaining: 2.04s
211:    learn: 0.2644634        total: 548ms    remaining: 2.04s
212:    learn: 0.2642684        total: 551ms    remaining: 2.03s
213:    learn: 0.2639989        total: 553ms    remaining: 2.03s
214:    learn: 0.2636934        total: 556ms    remaining: 2.03s
215:    learn: 0.2635280        total: 558ms    remaining: 2.02s
216:    learn: 0.2633429        total: 561ms    remaining: 2.02s
217:    learn: 0.2631460        total: 563ms    remaining: 2.02s
```

```
218:    learn: 0.2630033        total: 566ms    remaining: 2.02s
219:    learn: 0.2627488        total: 568ms    remaining: 2.02s
220:    learn: 0.2624358        total: 571ms    remaining: 2.01s
221:    learn: 0.2622292        total: 573ms    remaining: 2.01s
222:    learn: 0.2620083        total: 576ms    remaining: 2.01s
223:    learn: 0.2618307        total: 578ms    remaining: 2s
224:    learn: 0.2616477        total: 581ms    remaining: 2s
225:    learn: 0.2613978        total: 584ms    remaining: 2s
226:    learn: 0.2611139        total: 586ms    remaining: 2s
227:    learn: 0.2609467        total: 589ms    remaining: 1.99s
228:    learn: 0.2607375        total: 591ms    remaining: 1.99s
229:    learn: 0.2605113        total: 594ms    remaining: 1.99s
230:    learn: 0.2602421        total: 596ms    remaining: 1.99s
231:    learn: 0.2601371        total: 599ms    remaining: 1.98s
232:    learn: 0.2600159        total: 602ms    remaining: 1.98s
233:    learn: 0.2598866        total: 605ms    remaining: 1.98s
234:    learn: 0.2596444        total: 607ms    remaining: 1.98s
235:    learn: 0.2593798        total: 610ms    remaining: 1.97s
236:    learn: 0.2590858        total: 612ms    remaining: 1.97s
237:    learn: 0.2589325        total: 615ms    remaining: 1.97s
238:    learn: 0.2587623        total: 618ms    remaining: 1.97s
239:    learn: 0.2585421        total: 620ms    remaining: 1.96s
240:    learn: 0.2583588        total: 623ms    remaining: 1.96s
241:    learn: 0.2582033        total: 626ms    remaining: 1.96s
242:    learn: 0.2580732        total: 628ms    remaining: 1.96s
243:    learn: 0.2577596        total: 631ms    remaining: 1.96s
244:    learn: 0.2575093        total: 633ms    remaining: 1.95s
245:    learn: 0.2573064        total: 636ms    remaining: 1.95s
246:    learn: 0.2570572        total: 638ms    remaining: 1.95s
247:    learn: 0.2569246        total: 641ms    remaining: 1.94s
248:    learn: 0.2566989        total: 643ms    remaining: 1.94s
249:    learn: 0.2565134        total: 646ms    remaining: 1.94s
250:    learn: 0.2562937        total: 648ms    remaining: 1.93s
251:    learn: 0.2561428        total: 651ms    remaining: 1.93s
252:    learn: 0.2560519        total: 653ms    remaining: 1.93s
253:    learn: 0.2560046        total: 655ms    remaining: 1.92s
254:    learn: 0.2558513        total: 657ms    remaining: 1.92s
255:    learn: 0.2556419        total: 660ms    remaining: 1.92s
256:    learn: 0.2553758        total: 662ms    remaining: 1.91s
257:    learn: 0.2552515        total: 665ms    remaining: 1.91s
258:    learn: 0.2551243        total: 668ms    remaining: 1.91s
259:    learn: 0.2550429        total: 670ms    remaining: 1.91s
260:    learn: 0.2548059        total: 672ms    remaining: 1.9s
261:    learn: 0.2546698        total: 675ms    remaining: 1.9s
262:    learn: 0.2544298        total: 678ms    remaining: 1.9s
263:    learn: 0.2541459        total: 680ms    remaining: 1.9s
264:    learn: 0.2540021        total: 683ms    remaining: 1.89s
265:    learn: 0.2538532        total: 686ms    remaining: 1.89s
```

```
266:    learn: 0.2535521        total: 688ms    remaining: 1.89s
267:    learn: 0.2533359        total: 691ms    remaining: 1.89s
268:    learn: 0.2531032        total: 693ms    remaining: 1.88s
269:    learn: 0.2529227        total: 696ms    remaining: 1.88s
270:    learn: 0.2527160        total: 700ms    remaining: 1.88s
271:    learn: 0.2524820        total: 703ms    remaining: 1.88s
272:    learn: 0.2523279        total: 705ms    remaining: 1.88s
273:    learn: 0.2521617        total: 709ms    remaining: 1.88s
274:    learn: 0.2520830        total: 711ms    remaining: 1.87s
275:    learn: 0.2519324        total: 714ms    remaining: 1.87s
276:    learn: 0.2518630        total: 717ms    remaining: 1.87s
277:    learn: 0.2516161        total: 720ms    remaining: 1.87s
278:    learn: 0.2514079        total: 722ms    remaining: 1.87s
279:    learn: 0.2513188        total: 725ms    remaining: 1.86s
280:    learn: 0.2511217        total: 728ms    remaining: 1.86s
281:    learn: 0.2509871        total: 730ms    remaining: 1.86s
282:    learn: 0.2508467        total: 733ms    remaining: 1.85s
283:    learn: 0.2507036        total: 735ms    remaining: 1.85s
284:    learn: 0.2505327        total: 738ms    remaining: 1.85s
285:    learn: 0.2503316        total: 740ms    remaining: 1.85s
286:    learn: 0.2502110        total: 743ms    remaining: 1.84s
287:    learn: 0.2499857        total: 745ms    remaining: 1.84s
288:    learn: 0.2496659        total: 747ms    remaining: 1.84s
289:    learn: 0.2493982        total: 750ms    remaining: 1.83s
290:    learn: 0.2492811        total: 752ms    remaining: 1.83s
291:    learn: 0.2490959        total: 754ms    remaining: 1.83s
292:    learn: 0.2489460        total: 757ms    remaining: 1.83s
293:    learn: 0.2486826        total: 759ms    remaining: 1.82s
294:    learn: 0.2485505        total: 762ms    remaining: 1.82s
295:    learn: 0.2483789        total: 764ms    remaining: 1.82s
296:    learn: 0.2482668        total: 767ms    remaining: 1.81s
297:    learn: 0.2481358        total: 770ms    remaining: 1.81s
298:    learn: 0.2478984        total: 772ms    remaining: 1.81s
299:    learn: 0.2477825        total: 775ms    remaining: 1.81s
300:    learn: 0.2476463        total: 777ms    remaining: 1.8s
301:    learn: 0.2474000        total: 780ms    remaining: 1.8s
302:    learn: 0.2473103        total: 784ms    remaining: 1.8s
303:    learn: 0.2471715        total: 787ms    remaining: 1.8s
304:    learn: 0.2470133        total: 790ms    remaining: 1.8s
305:    learn: 0.2468311        total: 793ms    remaining: 1.8s
306:    learn: 0.2466344        total: 795ms    remaining: 1.79s
307:    learn: 0.2463371        total: 798ms    remaining: 1.79s
308:    learn: 0.2462387        total: 800ms    remaining: 1.79s
309:    learn: 0.2460780        total: 803ms    remaining: 1.79s
310:    learn: 0.2458055        total: 805ms    remaining: 1.78s
311:    learn: 0.2456757        total: 808ms    remaining: 1.78s
312:    learn: 0.2454557        total: 810ms    remaining: 1.78s
313:    learn: 0.2453402        total: 813ms    remaining: 1.77s
```

```
314:    learn: 0.2450563        total: 815ms    remaining: 1.77s
315:    learn: 0.2449059        total: 817ms    remaining: 1.77s
316:    learn: 0.2446746        total: 820ms    remaining: 1.77s
317:    learn: 0.2444367        total: 823ms    remaining: 1.76s
318:    learn: 0.2442537        total: 826ms    remaining: 1.76s
319:    learn: 0.2441146        total: 828ms    remaining: 1.76s
320:    learn: 0.2438167        total: 831ms    remaining: 1.76s
321:    learn: 0.2436942        total: 833ms    remaining: 1.75s
322:    learn: 0.2435757        total: 835ms    remaining: 1.75s
323:    learn: 0.2433447        total: 837ms    remaining: 1.75s
324:    learn: 0.2432207        total: 839ms    remaining: 1.74s
325:    learn: 0.2431057        total: 842ms    remaining: 1.74s
326:    learn: 0.2429948        total: 844ms    remaining: 1.74s
327:    learn: 0.2427614        total: 847ms    remaining: 1.74s
328:    learn: 0.2426304        total: 849ms    remaining: 1.73s
329:    learn: 0.2425232        total: 852ms    remaining: 1.73s
330:    learn: 0.2422891        total: 854ms    remaining: 1.73s
331:    learn: 0.2421477        total: 856ms    remaining: 1.72s
332:    learn: 0.2419581        total: 859ms    remaining: 1.72s
333:    learn: 0.2418237        total: 861ms    remaining: 1.72s
334:    learn: 0.2417355        total: 864ms    remaining: 1.71s
335:    learn: 0.2415198        total: 867ms    remaining: 1.71s
336:    learn: 0.2414081        total: 870ms    remaining: 1.71s
337:    learn: 0.2412678        total: 872ms    remaining: 1.71s
338:    learn: 0.2412015        total: 875ms    remaining: 1.71s
339:    learn: 0.2409755        total: 877ms    remaining: 1.7s
340:    learn: 0.2408700        total: 879ms    remaining: 1.7s
341:    learn: 0.2406700        total: 882ms    remaining: 1.7s
342:    learn: 0.2405434        total: 884ms    remaining: 1.69s
343:    learn: 0.2403597        total: 887ms    remaining: 1.69s
344:    learn: 0.2402093        total: 889ms    remaining: 1.69s
345:    learn: 0.2400890        total: 892ms    remaining: 1.69s
346:    learn: 0.2399819        total: 894ms    remaining: 1.68s
347:    learn: 0.2397973        total: 897ms    remaining: 1.68s
348:    learn: 0.2396115        total: 899ms    remaining: 1.68s
349:    learn: 0.2394743        total: 902ms    remaining: 1.67s
350:    learn: 0.2392951        total: 904ms    remaining: 1.67s
351:    learn: 0.2391568        total: 907ms    remaining: 1.67s
352:    learn: 0.2389358        total: 909ms    remaining: 1.67s
353:    learn: 0.2387980        total: 912ms    remaining: 1.66s
354:    learn: 0.2387952        total: 913ms    remaining: 1.66s
355:    learn: 0.2386995        total: 916ms    remaining: 1.66s
356:    learn: 0.2384815        total: 919ms    remaining: 1.66s
357:    learn: 0.2383320        total: 922ms    remaining: 1.65s
358:    learn: 0.2381440        total: 924ms    remaining: 1.65s
359:    learn: 0.2379944        total: 927ms    remaining: 1.65s
360:    learn: 0.2378133        total: 929ms    remaining: 1.65s
361:    learn: 0.2375872        total: 932ms    remaining: 1.64s
```

```
362:    learn: 0.2374678        total: 935ms    remaining: 1.64s
363:    learn: 0.2374142        total: 938ms    remaining: 1.64s
364:    learn: 0.2372606        total: 940ms    remaining: 1.64s
365:    learn: 0.2371268        total: 942ms    remaining: 1.63s
366:    learn: 0.2369932        total: 945ms    remaining: 1.63s
367:    learn: 0.2367728        total: 948ms    remaining: 1.63s
368:    learn: 0.2365443        total: 950ms    remaining: 1.62s
369:    learn: 0.2363600        total: 953ms    remaining: 1.62s
370:    learn: 0.2362136        total: 956ms    remaining: 1.62s
371:    learn: 0.2360867        total: 959ms    remaining: 1.62s
372:    learn: 0.2359681        total: 961ms    remaining: 1.61s
373:    learn: 0.2357716        total: 963ms    remaining: 1.61s
374:    learn: 0.2355763        total: 966ms    remaining: 1.61s
375:    learn: 0.2353696        total: 968ms    remaining: 1.61s
376:    learn: 0.2351695        total: 971ms    remaining: 1.6s
377:    learn: 0.2350320        total: 974ms    remaining: 1.6s
378:    learn: 0.2348695        total: 977ms    remaining: 1.6s
379:    learn: 0.2347497        total: 979ms    remaining: 1.6s
380:    learn: 0.2345120        total: 982ms    remaining: 1.59s
381:    learn: 0.2343889        total: 984ms    remaining: 1.59s
382:    learn: 0.2342061        total: 987ms    remaining: 1.59s
383:    learn: 0.2340127        total: 989ms    remaining: 1.59s
384:    learn: 0.2338713        total: 991ms    remaining: 1.58s
385:    learn: 0.2336900        total: 994ms    remaining: 1.58s
386:    learn: 0.2335714        total: 996ms    remaining: 1.58s
387:    learn: 0.2333205        total: 999ms    remaining: 1.57s
388:    learn: 0.2331754        total: 1s       remaining: 1.57s
389:    learn: 0.2330078        total: 1s       remaining: 1.57s
390:    learn: 0.2328771        total: 1.01s    remaining: 1.57s
391:    learn: 0.2327749        total: 1.01s    remaining: 1.57s
392:    learn: 0.2326453        total: 1.01s    remaining: 1.56s
393:    learn: 0.2324682        total: 1.01s    remaining: 1.56s
394:    learn: 0.2323467        total: 1.02s    remaining: 1.56s
395:    learn: 0.2321787        total: 1.02s    remaining: 1.56s
396:    learn: 0.2320511        total: 1.02s    remaining: 1.55s
397:    learn: 0.2319083        total: 1.03s    remaining: 1.55s
398:    learn: 0.2317418        total: 1.03s    remaining: 1.55s
399:    learn: 0.2315052        total: 1.03s    remaining: 1.55s
400:    learn: 0.2313389        total: 1.03s    remaining: 1.54s
401:    learn: 0.2312412        total: 1.03s    remaining: 1.54s
402:    learn: 0.2311488        total: 1.04s    remaining: 1.54s
403:    learn: 0.2310485        total: 1.04s    remaining: 1.53s
404:    learn: 0.2309377        total: 1.04s    remaining: 1.53s
405:    learn: 0.2307222        total: 1.04s    remaining: 1.53s
406:    learn: 0.2306021        total: 1.05s    remaining: 1.53s
407:    learn: 0.2305027        total: 1.05s    remaining: 1.53s
408:    learn: 0.2303061        total: 1.05s    remaining: 1.52s
409:    learn: 0.2301313        total: 1.06s    remaining: 1.52s
```

```
410:    learn: 0.2300177        total: 1.06s    remaining: 1.52s
411:    learn: 0.2298662        total: 1.06s    remaining: 1.52s
412:    learn: 0.2297217        total: 1.06s    remaining: 1.51s
413:    learn: 0.2295304        total: 1.07s    remaining: 1.51s
414:    learn: 0.2293233        total: 1.07s    remaining: 1.51s
415:    learn: 0.2291539        total: 1.07s    remaining: 1.51s
416:    learn: 0.2289438        total: 1.07s    remaining: 1.5s
417:    learn: 0.2287930        total: 1.08s    remaining: 1.5s
418:    learn: 0.2286929        total: 1.08s    remaining: 1.5s
419:    learn: 0.2285816        total: 1.08s    remaining: 1.5s
420:    learn: 0.2283660        total: 1.08s    remaining: 1.49s
421:    learn: 0.2282691        total: 1.09s    remaining: 1.49s
422:    learn: 0.2281276        total: 1.09s    remaining: 1.49s
423:    learn: 0.2278513        total: 1.09s    remaining: 1.48s
424:    learn: 0.2277520        total: 1.09s    remaining: 1.48s
425:    learn: 0.2276181        total: 1.1s     remaining: 1.48s
426:    learn: 0.2274697        total: 1.1s     remaining: 1.48s
427:    learn: 0.2272820        total: 1.1s     remaining: 1.47s
428:    learn: 0.2272038        total: 1.1s     remaining: 1.47s
429:    learn: 0.2270718        total: 1.11s    remaining: 1.47s
430:    learn: 0.2269545        total: 1.11s    remaining: 1.47s
431:    learn: 0.2267885        total: 1.11s    remaining: 1.46s
432:    learn: 0.2266109        total: 1.11s    remaining: 1.46s
433:    learn: 0.2264600        total: 1.12s    remaining: 1.46s
434:    learn: 0.2262436        total: 1.12s    remaining: 1.46s
435:    learn: 0.2261456        total: 1.12s    remaining: 1.45s
436:    learn: 0.2260495        total: 1.13s    remaining: 1.45s
437:    learn: 0.2258593        total: 1.13s    remaining: 1.45s
438:    learn: 0.2255212        total: 1.13s    remaining: 1.45s
439:    learn: 0.2254207        total: 1.13s    remaining: 1.44s
440:    learn: 0.2252927        total: 1.14s    remaining: 1.44s
441:    learn: 0.2252084        total: 1.14s    remaining: 1.44s
442:    learn: 0.2250267        total: 1.14s    remaining: 1.44s
443:    learn: 0.2248708        total: 1.15s    remaining: 1.44s
444:    learn: 0.2247245        total: 1.15s    remaining: 1.43s
445:    learn: 0.2246120        total: 1.15s    remaining: 1.43s
446:    learn: 0.2244545        total: 1.15s    remaining: 1.43s
447:    learn: 0.2242356        total: 1.16s    remaining: 1.43s
448:    learn: 0.2239810        total: 1.16s    remaining: 1.42s
449:    learn: 0.2238682        total: 1.16s    remaining: 1.42s
450:    learn: 0.2237306        total: 1.17s    remaining: 1.42s
451:    learn: 0.2235472        total: 1.17s    remaining: 1.42s
452:    learn: 0.2234455        total: 1.17s    remaining: 1.41s
453:    learn: 0.2232335        total: 1.17s    remaining: 1.41s
454:    learn: 0.2230796        total: 1.18s    remaining: 1.41s
455:    learn: 0.2229195        total: 1.18s    remaining: 1.41s
456:    learn: 0.2227765        total: 1.18s    remaining: 1.4s
457:    learn: 0.2225850        total: 1.18s    remaining: 1.4s
```

```
458:    learn: 0.2224444        total: 1.19s    remaining: 1.4s
459:    learn: 0.2222827        total: 1.19s    remaining: 1.4s
460:    learn: 0.2221966        total: 1.19s    remaining: 1.39s
461:    learn: 0.2220944        total: 1.19s    remaining: 1.39s
462:    learn: 0.2219213        total: 1.2s     remaining: 1.39s
463:    learn: 0.2218407        total: 1.2s     remaining: 1.38s
464:    learn: 0.2217284        total: 1.2s     remaining: 1.38s
465:    learn: 0.2216004        total: 1.2s     remaining: 1.38s
466:    learn: 0.2214784        total: 1.21s    remaining: 1.38s
467:    learn: 0.2212879        total: 1.21s    remaining: 1.37s
468:    learn: 0.2211433        total: 1.21s    remaining: 1.37s
469:    learn: 0.2209757        total: 1.21s    remaining: 1.37s
470:    learn: 0.2208689        total: 1.22s    remaining: 1.37s
471:    learn: 0.2207924        total: 1.22s    remaining: 1.36s
472:    learn: 0.2205848        total: 1.22s    remaining: 1.36s
473:    learn: 0.2204481        total: 1.22s    remaining: 1.36s
474:    learn: 0.2202680        total: 1.23s    remaining: 1.36s
475:    learn: 0.2201687        total: 1.23s    remaining: 1.35s
476:    learn: 0.2200259        total: 1.23s    remaining: 1.35s
477:    learn: 0.2199538        total: 1.23s    remaining: 1.35s
478:    learn: 0.2198262        total: 1.24s    remaining: 1.34s
479:    learn: 0.2197696        total: 1.24s    remaining: 1.34s
480:    learn: 0.2195758        total: 1.24s    remaining: 1.34s
481:    learn: 0.2194119        total: 1.24s    remaining: 1.33s
482:    learn: 0.2192567        total: 1.24s    remaining: 1.33s
483:    learn: 0.2189899        total: 1.25s    remaining: 1.33s
484:    learn: 0.2187136        total: 1.25s    remaining: 1.33s
485:    learn: 0.2185014        total: 1.25s    remaining: 1.32s
486:    learn: 0.2183326        total: 1.25s    remaining: 1.32s
487:    learn: 0.2182379        total: 1.26s    remaining: 1.32s
488:    learn: 0.2181379        total: 1.26s    remaining: 1.32s
489:    learn: 0.2180178        total: 1.26s    remaining: 1.31s
490:    learn: 0.2178813        total: 1.26s    remaining: 1.31s
491:    learn: 0.2177714        total: 1.27s    remaining: 1.31s
492:    learn: 0.2175406        total: 1.27s    remaining: 1.31s
493:    learn: 0.2174223        total: 1.27s    remaining: 1.3s
494:    learn: 0.2172845        total: 1.27s    remaining: 1.3s
495:    learn: 0.2171484        total: 1.28s    remaining: 1.3s
496:    learn: 0.2170407        total: 1.28s    remaining: 1.29s
497:    learn: 0.2168615        total: 1.28s    remaining: 1.29s
498:    learn: 0.2167404        total: 1.28s    remaining: 1.29s
499:    learn: 0.2165757        total: 1.29s    remaining: 1.29s
500:    learn: 0.2163450        total: 1.29s    remaining: 1.28s
501:    learn: 0.2162640        total: 1.29s    remaining: 1.28s
502:    learn: 0.2161811        total: 1.29s    remaining: 1.28s
503:    learn: 0.2160729        total: 1.3s     remaining: 1.27s
504:    learn: 0.2159763        total: 1.3s     remaining: 1.27s
505:    learn: 0.2158867        total: 1.3s     remaining: 1.27s
```

```
506:    learn: 0.2158339        total: 1.3s     remaining: 1.27s
507:    learn: 0.2157056        total: 1.31s    remaining: 1.26s
508:    learn: 0.2155437        total: 1.31s    remaining: 1.26s
509:    learn: 0.2154609        total: 1.31s    remaining: 1.26s
510:    learn: 0.2153518        total: 1.31s    remaining: 1.26s
511:    learn: 0.2152186        total: 1.32s    remaining: 1.25s
512:    learn: 0.2151123        total: 1.32s    remaining: 1.25s
513:    learn: 0.2149629        total: 1.32s    remaining: 1.25s
514:    learn: 0.2148171        total: 1.32s    remaining: 1.25s
515:    learn: 0.2147382        total: 1.33s    remaining: 1.25s
516:    learn: 0.2146506        total: 1.33s    remaining: 1.24s
517:    learn: 0.2144189        total: 1.33s    remaining: 1.24s
518:    learn: 0.2142324        total: 1.33s    remaining: 1.24s
519:    learn: 0.2141139        total: 1.34s    remaining: 1.24s
520:    learn: 0.2139571        total: 1.34s    remaining: 1.23s
521:    learn: 0.2138883        total: 1.34s    remaining: 1.23s
522:    learn: 0.2138282        total: 1.34s    remaining: 1.23s
523:    learn: 0.2137720        total: 1.35s    remaining: 1.22s
524:    learn: 0.2136508        total: 1.35s    remaining: 1.22s
525:    learn: 0.2135551        total: 1.35s    remaining: 1.22s
526:    learn: 0.2134267        total: 1.35s    remaining: 1.22s
527:    learn: 0.2132987        total: 1.36s    remaining: 1.21s
528:    learn: 0.2131713        total: 1.36s    remaining: 1.21s
529:    learn: 0.2130620        total: 1.36s    remaining: 1.21s
530:    learn: 0.2129562        total: 1.36s    remaining: 1.2s
531:    learn: 0.2128931        total: 1.37s    remaining: 1.2s
532:    learn: 0.2127742        total: 1.37s    remaining: 1.2s
533:    learn: 0.2126590        total: 1.37s    remaining: 1.2s
534:    learn: 0.2125283        total: 1.38s    remaining: 1.2s
535:    learn: 0.2124269        total: 1.38s    remaining: 1.19s
536:    learn: 0.2122823        total: 1.38s    remaining: 1.19s
537:    learn: 0.2122051        total: 1.38s    remaining: 1.19s
538:    learn: 0.2120660        total: 1.39s    remaining: 1.19s
539:    learn: 0.2118419        total: 1.39s    remaining: 1.18s
540:    learn: 0.2117479        total: 1.39s    remaining: 1.18s
541:    learn: 0.2116091        total: 1.39s    remaining: 1.18s
542:    learn: 0.2114257        total: 1.4s     remaining: 1.18s
543:    learn: 0.2113058        total: 1.4s     remaining: 1.17s
544:    learn: 0.2112383        total: 1.4s     remaining: 1.17s
545:    learn: 0.2111530        total: 1.4s     remaining: 1.17s
546:    learn: 0.2109904        total: 1.41s    remaining: 1.16s
547:    learn: 0.2108715        total: 1.41s    remaining: 1.16s
548:    learn: 0.2107601        total: 1.41s    remaining: 1.16s
549:    learn: 0.2105909        total: 1.41s    remaining: 1.16s
550:    learn: 0.2104616        total: 1.42s    remaining: 1.15s
551:    learn: 0.2103956        total: 1.42s    remaining: 1.15s
552:    learn: 0.2102114        total: 1.42s    remaining: 1.15s
553:    learn: 0.2101000        total: 1.42s    remaining: 1.15s
```

```
554:    learn: 0.2099882        total: 1.43s    remaining: 1.14s
555:    learn: 0.2098634        total: 1.43s    remaining: 1.14s
556:    learn: 0.2097737        total: 1.43s    remaining: 1.14s
557:    learn: 0.2097073        total: 1.43s    remaining: 1.14s
558:    learn: 0.2095127        total: 1.44s    remaining: 1.13s
559:    learn: 0.2093714        total: 1.44s    remaining: 1.13s
560:    learn: 0.2092298        total: 1.44s    remaining: 1.13s
561:    learn: 0.2091277        total: 1.44s    remaining: 1.13s
562:    learn: 0.2090535        total: 1.45s    remaining: 1.12s
563:    learn: 0.2088876        total: 1.45s    remaining: 1.12s
564:    learn: 0.2087073        total: 1.45s    remaining: 1.12s
565:    learn: 0.2085431        total: 1.45s    remaining: 1.11s
566:    learn: 0.2084329        total: 1.46s    remaining: 1.11s
567:    learn: 0.2083777        total: 1.46s    remaining: 1.11s
568:    learn: 0.2082463        total: 1.46s    remaining: 1.11s
569:    learn: 0.2081353        total: 1.46s    remaining: 1.1s
570:    learn: 0.2080080        total: 1.47s    remaining: 1.1s
571:    learn: 0.2079417        total: 1.47s    remaining: 1.1s
572:    learn: 0.2078150        total: 1.47s    remaining: 1.1s
573:    learn: 0.2077468        total: 1.47s    remaining: 1.09s
574:    learn: 0.2076407        total: 1.48s    remaining: 1.09s
575:    learn: 0.2075091        total: 1.48s    remaining: 1.09s
576:    learn: 0.2073973        total: 1.48s    remaining: 1.09s
577:    learn: 0.2073365        total: 1.48s    remaining: 1.08s
578:    learn: 0.2072507        total: 1.49s    remaining: 1.08s
579:    learn: 0.2071850        total: 1.49s    remaining: 1.08s
580:    learn: 0.2070861        total: 1.49s    remaining: 1.07s
581:    learn: 0.2069900        total: 1.49s    remaining: 1.07s
582:    learn: 0.2068491        total: 1.5s     remaining: 1.07s
583:    learn: 0.2067478        total: 1.5s     remaining: 1.07s
584:    learn: 0.2066594        total: 1.5s     remaining: 1.06s
585:    learn: 0.2065534        total: 1.5s     remaining: 1.06s
586:    learn: 0.2064145        total: 1.51s    remaining: 1.06s
587:    learn: 0.2063097        total: 1.51s    remaining: 1.06s
588:    learn: 0.2062434        total: 1.51s    remaining: 1.05s
589:    learn: 0.2061579        total: 1.52s    remaining: 1.05s
590:    learn: 0.2060570        total: 1.52s    remaining: 1.05s
591:    learn: 0.2058879        total: 1.52s    remaining: 1.05s
592:    learn: 0.2056247        total: 1.52s    remaining: 1.05s
593:    learn: 0.2055301        total: 1.53s    remaining: 1.04s
594:    learn: 0.2054353        total: 1.53s    remaining: 1.04s
595:    learn: 0.2052964        total: 1.53s    remaining: 1.04s
596:    learn: 0.2051851        total: 1.54s    remaining: 1.04s
597:    learn: 0.2050567        total: 1.54s    remaining: 1.03s
598:    learn: 0.2049185        total: 1.54s    remaining: 1.03s
599:    learn: 0.2047526        total: 1.54s    remaining: 1.03s
600:    learn: 0.2046471        total: 1.55s    remaining: 1.03s
601:    learn: 0.2045188        total: 1.55s    remaining: 1.02s
```

```
602:    learn: 0.2044281        total: 1.55s    remaining: 1.02s
603:    learn: 0.2043544        total: 1.55s    remaining: 1.02s
604:    learn: 0.2042749        total: 1.56s    remaining: 1.02s
605:    learn: 0.2042219        total: 1.56s    remaining: 1.01s
606:    learn: 0.2041358        total: 1.56s    remaining: 1.01s
607:    learn: 0.2040525        total: 1.56s    remaining: 1.01s
608:    learn: 0.2039098        total: 1.57s    remaining: 1.01s
609:    learn: 0.2038396        total: 1.57s    remaining: 1s
610:    learn: 0.2037486        total: 1.57s    remaining: 1s
611:    learn: 0.2036848        total: 1.57s    remaining: 998ms
612:    learn: 0.2036317        total: 1.58s    remaining: 996ms
613:    learn: 0.2035275        total: 1.58s    remaining: 993ms
614:    learn: 0.2034413        total: 1.58s    remaining: 990ms
615:    learn: 0.2033958        total: 1.58s    remaining: 988ms
616:    learn: 0.2032446        total: 1.59s    remaining: 985ms
617:    learn: 0.2031640        total: 1.59s    remaining: 983ms
618:    learn: 0.2030587        total: 1.59s    remaining: 980ms
619:    learn: 0.2029653        total: 1.59s    remaining: 977ms
620:    learn: 0.2028029        total: 1.6s     remaining: 975ms
621:    learn: 0.2026717        total: 1.6s     remaining: 972ms
622:    learn: 0.2025496        total: 1.6s     remaining: 970ms
623:    learn: 0.2024364        total: 1.6s     remaining: 967ms
624:    learn: 0.2023467        total: 1.61s    remaining: 964ms
625:    learn: 0.2021886        total: 1.61s    remaining: 962ms
626:    learn: 0.2020898        total: 1.61s    remaining: 959ms
627:    learn: 0.2020395        total: 1.61s    remaining: 957ms
628:    learn: 0.2018959        total: 1.62s    remaining: 954ms
629:    learn: 0.2018369        total: 1.62s    remaining: 951ms
630:    learn: 0.2017329        total: 1.62s    remaining: 949ms
631:    learn: 0.2016360        total: 1.63s    remaining: 946ms
632:    learn: 0.2015279        total: 1.63s    remaining: 944ms
633:    learn: 0.2014382        total: 1.63s    remaining: 941ms
634:    learn: 0.2012785        total: 1.63s    remaining: 938ms
635:    learn: 0.2012020        total: 1.64s    remaining: 936ms
636:    learn: 0.2010486        total: 1.64s    remaining: 933ms
637:    learn: 0.2009951        total: 1.64s    remaining: 931ms
638:    learn: 0.2009279        total: 1.64s    remaining: 928ms
639:    learn: 0.2007737        total: 1.65s    remaining: 925ms
640:    learn: 0.2006652        total: 1.65s    remaining: 923ms
641:    learn: 0.2005113        total: 1.65s    remaining: 920ms
642:    learn: 0.2004343        total: 1.65s    remaining: 918ms
643:    learn: 0.2003371        total: 1.66s    remaining: 915ms
644:    learn: 0.2002373        total: 1.66s    remaining: 912ms
645:    learn: 0.2001048        total: 1.66s    remaining: 910ms
646:    learn: 0.1999415        total: 1.66s    remaining: 907ms
647:    learn: 0.1998555        total: 1.67s    remaining: 905ms
648:    learn: 0.1997285        total: 1.67s    remaining: 902ms
649:    learn: 0.1996727        total: 1.67s    remaining: 900ms
```

```
650:    learn: 0.1995137        total: 1.67s    remaining: 897ms
651:    learn: 0.1994409        total: 1.68s    remaining: 895ms
652:    learn: 0.1993693        total: 1.68s    remaining: 892ms
653:    learn: 0.1992055        total: 1.68s    remaining: 889ms
654:    learn: 0.1991202        total: 1.68s    remaining: 887ms
655:    learn: 0.1990637        total: 1.69s    remaining: 884ms
656:    learn: 0.1990071        total: 1.69s    remaining: 881ms
657:    learn: 0.1989391        total: 1.69s    remaining: 879ms
658:    learn: 0.1988775        total: 1.69s    remaining: 876ms
659:    learn: 0.1987468        total: 1.7s     remaining: 873ms
660:    learn: 0.1986693        total: 1.7s     remaining: 871ms
661:    learn: 0.1985528        total: 1.7s     remaining: 868ms
662:    learn: 0.1983925        total: 1.7s     remaining: 866ms
663:    learn: 0.1983055        total: 1.71s    remaining: 863ms
664:    learn: 0.1981598        total: 1.71s    remaining: 861ms
665:    learn: 0.1980481        total: 1.71s    remaining: 859ms
666:    learn: 0.1979907        total: 1.71s    remaining: 856ms
667:    learn: 0.1978288        total: 1.72s    remaining: 853ms
668:    learn: 0.1977012        total: 1.72s    remaining: 851ms
669:    learn: 0.1975355        total: 1.72s    remaining: 848ms
670:    learn: 0.1973942        total: 1.73s    remaining: 846ms
671:    learn: 0.1973025        total: 1.73s    remaining: 843ms
672:    learn: 0.1971503        total: 1.73s    remaining: 841ms
673:    learn: 0.1971009        total: 1.73s    remaining: 838ms
674:    learn: 0.1969734        total: 1.74s    remaining: 836ms
675:    learn: 0.1967857        total: 1.74s    remaining: 833ms
676:    learn: 0.1966838        total: 1.74s    remaining: 830ms
677:    learn: 0.1965502        total: 1.74s    remaining: 828ms
678:    learn: 0.1964339        total: 1.75s    remaining: 825ms
679:    learn: 0.1963307        total: 1.75s    remaining: 822ms
680:    learn: 0.1962457        total: 1.75s    remaining: 820ms
681:    learn: 0.1960196        total: 1.75s    remaining: 817ms
682:    learn: 0.1958964        total: 1.75s    remaining: 814ms
683:    learn: 0.1958108        total: 1.76s    remaining: 812ms
684:    learn: 0.1957445        total: 1.76s    remaining: 809ms
685:    learn: 0.1956851        total: 1.76s    remaining: 807ms
686:    learn: 0.1955620        total: 1.76s    remaining: 804ms
687:    learn: 0.1954663        total: 1.77s    remaining: 801ms
688:    learn: 0.1953814        total: 1.77s    remaining: 799ms
689:    learn: 0.1952834        total: 1.77s    remaining: 796ms
690:    learn: 0.1951326        total: 1.77s    remaining: 793ms
691:    learn: 0.1949874        total: 1.78s    remaining: 791ms
692:    learn: 0.1948937        total: 1.78s    remaining: 788ms
693:    learn: 0.1948094        total: 1.78s    remaining: 785ms
694:    learn: 0.1947100        total: 1.78s    remaining: 783ms
695:    learn: 0.1945462        total: 1.78s    remaining: 780ms
696:    learn: 0.1944862        total: 1.79s    remaining: 777ms
697:    learn: 0.1943791        total: 1.79s    remaining: 774ms
```

```
698:    learn: 0.1943047        total: 1.79s    remaining: 771ms
699:    learn: 0.1941735        total: 1.79s    remaining: 769ms
700:    learn: 0.1940921        total: 1.8s     remaining: 766ms
701:    learn: 0.1939817        total: 1.8s     remaining: 764ms
702:    learn: 0.1938574        total: 1.8s     remaining: 761ms
703:    learn: 0.1937707        total: 1.8s     remaining: 759ms
704:    learn: 0.1935954        total: 1.81s    remaining: 756ms
705:    learn: 0.1934558        total: 1.81s    remaining: 753ms
706:    learn: 0.1933791        total: 1.81s    remaining: 751ms
707:    learn: 0.1931993        total: 1.81s    remaining: 748ms
708:    learn: 0.1931398        total: 1.82s    remaining: 745ms
709:    learn: 0.1930236        total: 1.82s    remaining: 743ms
710:    learn: 0.1929174        total: 1.82s    remaining: 740ms
711:    learn: 0.1928323        total: 1.82s    remaining: 737ms
712:    learn: 0.1927509        total: 1.82s    remaining: 735ms
713:    learn: 0.1926172        total: 1.83s    remaining: 732ms
714:    learn: 0.1925231        total: 1.83s    remaining: 730ms
715:    learn: 0.1923729        total: 1.83s    remaining: 727ms
716:    learn: 0.1922424        total: 1.83s    remaining: 724ms
717:    learn: 0.1921303        total: 1.84s    remaining: 722ms
718:    learn: 0.1919879        total: 1.84s    remaining: 719ms
719:    learn: 0.1919351        total: 1.84s    remaining: 717ms
720:    learn: 0.1918097        total: 1.84s    remaining: 714ms
721:    learn: 0.1917466        total: 1.85s    remaining: 711ms
722:    learn: 0.1916527        total: 1.85s    remaining: 709ms
723:    learn: 0.1915383        total: 1.85s    remaining: 707ms
724:    learn: 0.1914996        total: 1.86s    remaining: 704ms
725:    learn: 0.1914153        total: 1.86s    remaining: 702ms
726:    learn: 0.1912269        total: 1.86s    remaining: 699ms
727:    learn: 0.1911789        total: 1.86s    remaining: 696ms
728:    learn: 0.1910427        total: 1.87s    remaining: 694ms
729:    learn: 0.1910056        total: 1.87s    remaining: 691ms
730:    learn: 0.1908414        total: 1.87s    remaining: 689ms
731:    learn: 0.1907230        total: 1.87s    remaining: 686ms
732:    learn: 0.1906398        total: 1.88s    remaining: 683ms
733:    learn: 0.1905000        total: 1.88s    remaining: 681ms
734:    learn: 0.1904099        total: 1.88s    remaining: 678ms
735:    learn: 0.1903448        total: 1.88s    remaining: 675ms
736:    learn: 0.1902540        total: 1.89s    remaining: 673ms
737:    learn: 0.1901222        total: 1.89s    remaining: 670ms
738:    learn: 0.1899989        total: 1.89s    remaining: 667ms
739:    learn: 0.1899428        total: 1.89s    remaining: 664ms
740:    learn: 0.1898845        total: 1.89s    remaining: 662ms
741:    learn: 0.1898153        total: 1.9s     remaining: 659ms
742:    learn: 0.1896661        total: 1.9s     remaining: 656ms
743:    learn: 0.1895661        total: 1.9s     remaining: 653ms
744:    learn: 0.1895209        total: 1.9s     remaining: 650ms
745:    learn: 0.1893916        total: 1.9s     remaining: 648ms
```

```
746:    learn: 0.1892728        total: 1.9s     remaining: 645ms
747:    learn: 0.1891693        total: 1.91s    remaining: 643ms
748:    learn: 0.1890280        total: 1.91s    remaining: 640ms
749:    learn: 0.1888985        total: 1.91s    remaining: 637ms
750:    learn: 0.1887856        total: 1.91s    remaining: 635ms
751:    learn: 0.1886629        total: 1.92s    remaining: 632ms
752:    learn: 0.1885915        total: 1.92s    remaining: 630ms
753:    learn: 0.1885542        total: 1.92s    remaining: 627ms
754:    learn: 0.1884198        total: 1.92s    remaining: 625ms
755:    learn: 0.1882766        total: 1.93s    remaining: 622ms
756:    learn: 0.1882360        total: 1.93s    remaining: 620ms
757:    learn: 0.1881801        total: 1.93s    remaining: 617ms
758:    learn: 0.1880681        total: 1.94s    remaining: 615ms
759:    learn: 0.1879621        total: 1.94s    remaining: 612ms
760:    learn: 0.1878124        total: 1.94s    remaining: 609ms
761:    learn: 0.1877261        total: 1.94s    remaining: 607ms
762:    learn: 0.1876434        total: 1.95s    remaining: 604ms
763:    learn: 0.1875412        total: 1.95s    remaining: 602ms
764:    learn: 0.1874538        total: 1.95s    remaining: 599ms
765:    learn: 0.1873742        total: 1.95s    remaining: 597ms
766:    learn: 0.1873078        total: 1.96s    remaining: 594ms
767:    learn: 0.1872028        total: 1.96s    remaining: 591ms
768:    learn: 0.1870841        total: 1.96s    remaining: 589ms
769:    learn: 0.1869987        total: 1.96s    remaining: 586ms
770:    learn: 0.1868546        total: 1.97s    remaining: 584ms
771:    learn: 0.1867819        total: 1.97s    remaining: 581ms
772:    learn: 0.1866672        total: 1.97s    remaining: 578ms
773:    learn: 0.1865888        total: 1.97s    remaining: 576ms
774:    learn: 0.1865010        total: 1.97s    remaining: 573ms
775:    learn: 0.1863756        total: 1.98s    remaining: 571ms
776:    learn: 0.1862953        total: 1.98s    remaining: 568ms
777:    learn: 0.1861748        total: 1.98s    remaining: 566ms
778:    learn: 0.1861136        total: 1.98s    remaining: 563ms
779:    learn: 0.1860243        total: 1.99s    remaining: 561ms
780:    learn: 0.1859253        total: 1.99s    remaining: 558ms
781:    learn: 0.1858724        total: 1.99s    remaining: 556ms
782:    learn: 0.1857209        total: 2s       remaining: 553ms
783:    learn: 0.1855904        total: 2s       remaining: 551ms
784:    learn: 0.1854387        total: 2s       remaining: 548ms
785:    learn: 0.1853056        total: 2s       remaining: 546ms
786:    learn: 0.1851637        total: 2.01s    remaining: 543ms
787:    learn: 0.1850499        total: 2.01s    remaining: 540ms
788:    learn: 0.1849398        total: 2.01s    remaining: 538ms
789:    learn: 0.1847726        total: 2.01s    remaining: 535ms
790:    learn: 0.1846991        total: 2.02s    remaining: 533ms
791:    learn: 0.1845445        total: 2.02s    remaining: 530ms
792:    learn: 0.1844853        total: 2.02s    remaining: 528ms
793:    learn: 0.1843160        total: 2.02s    remaining: 525ms
```

```
794:    learn: 0.1841826        total: 2.03s    remaining: 522ms
795:    learn: 0.1841161        total: 2.03s    remaining: 520ms
796:    learn: 0.1839632        total: 2.03s    remaining: 517ms
797:    learn: 0.1839022        total: 2.03s    remaining: 515ms
798:    learn: 0.1838470        total: 2.04s    remaining: 512ms
799:    learn: 0.1837514        total: 2.04s    remaining: 510ms
800:    learn: 0.1836204        total: 2.04s    remaining: 507ms
801:    learn: 0.1835758        total: 2.04s    remaining: 505ms
802:    learn: 0.1834933        total: 2.04s    remaining: 502ms
803:    learn: 0.1833797        total: 2.05s    remaining: 499ms
804:    learn: 0.1832876        total: 2.05s    remaining: 497ms
805:    learn: 0.1832502        total: 2.05s    remaining: 494ms
806:    learn: 0.1831408        total: 2.06s    remaining: 492ms
807:    learn: 0.1830495        total: 2.06s    remaining: 489ms
808:    learn: 0.1829519        total: 2.06s    remaining: 487ms
809:    learn: 0.1828466        total: 2.06s    remaining: 484ms
810:    learn: 0.1827596        total: 2.06s    remaining: 481ms
811:    learn: 0.1826982        total: 2.07s    remaining: 479ms
812:    learn: 0.1826045        total: 2.07s    remaining: 476ms
813:    learn: 0.1824492        total: 2.07s    remaining: 474ms
814:    learn: 0.1823986        total: 2.08s    remaining: 471ms
815:    learn: 0.1822724        total: 2.08s    remaining: 469ms
816:    learn: 0.1821921        total: 2.08s    remaining: 466ms
817:    learn: 0.1820937        total: 2.08s    remaining: 464ms
818:    learn: 0.1819891        total: 2.09s    remaining: 461ms
819:    learn: 0.1818824        total: 2.09s    remaining: 459ms
820:    learn: 0.1818307        total: 2.09s    remaining: 456ms
821:    learn: 0.1817092        total: 2.09s    remaining: 453ms
822:    learn: 0.1816068        total: 2.1s     remaining: 451ms
823:    learn: 0.1815108        total: 2.1s     remaining: 448ms
824:    learn: 0.1814528        total: 2.1s     remaining: 446ms
825:    learn: 0.1813997        total: 2.1s     remaining: 443ms
826:    learn: 0.1812908        total: 2.11s    remaining: 441ms
827:    learn: 0.1812310        total: 2.11s    remaining: 438ms
828:    learn: 0.1811206        total: 2.11s    remaining: 436ms
829:    learn: 0.1810750        total: 2.11s    remaining: 433ms
830:    learn: 0.1810125        total: 2.12s    remaining: 430ms
831:    learn: 0.1809075        total: 2.12s    remaining: 428ms
832:    learn: 0.1808341        total: 2.12s    remaining: 425ms
833:    learn: 0.1807453        total: 2.12s    remaining: 423ms
834:    learn: 0.1806375        total: 2.13s    remaining: 420ms
835:    learn: 0.1805192        total: 2.13s    remaining: 417ms
836:    learn: 0.1804365        total: 2.13s    remaining: 415ms
837:    learn: 0.1803408        total: 2.13s    remaining: 412ms
838:    learn: 0.1802848        total: 2.13s    remaining: 410ms
839:    learn: 0.1802036        total: 2.14s    remaining: 407ms
840:    learn: 0.1801713        total: 2.14s    remaining: 405ms
841:    learn: 0.1800979        total: 2.14s    remaining: 402ms
```

```
842:    learn: 0.1800330        total: 2.14s    remaining: 399ms
843:    learn: 0.1799338        total: 2.15s    remaining: 397ms
844:    learn: 0.1798550        total: 2.15s    remaining: 394ms
845:    learn: 0.1797221        total: 2.15s    remaining: 392ms
846:    learn: 0.1796473        total: 2.15s    remaining: 389ms
847:    learn: 0.1795759        total: 2.16s    remaining: 387ms
848:    learn: 0.1794505        total: 2.16s    remaining: 384ms
849:    learn: 0.1793698        total: 2.16s    remaining: 382ms
850:    learn: 0.1792433        total: 2.17s    remaining: 379ms
851:    learn: 0.1792079        total: 2.17s    remaining: 377ms
852:    learn: 0.1791107        total: 2.17s    remaining: 374ms
853:    learn: 0.1790194        total: 2.17s    remaining: 372ms
854:    learn: 0.1789251        total: 2.17s    remaining: 369ms
855:    learn: 0.1788453        total: 2.18s    remaining: 366ms
856:    learn: 0.1787801        total: 2.18s    remaining: 364ms
857:    learn: 0.1786843        total: 2.18s    remaining: 361ms
858:    learn: 0.1785794        total: 2.19s    remaining: 359ms
859:    learn: 0.1784868        total: 2.19s    remaining: 356ms
860:    learn: 0.1784516        total: 2.19s    remaining: 354ms
861:    learn: 0.1783439        total: 2.19s    remaining: 351ms
862:    learn: 0.1782330        total: 2.19s    remaining: 349ms
863:    learn: 0.1781536        total: 2.2s     remaining: 346ms
864:    learn: 0.1780888        total: 2.2s     remaining: 343ms
865:    learn: 0.1779573        total: 2.2s     remaining: 341ms
866:    learn: 0.1778185        total: 2.21s    remaining: 338ms
867:    learn: 0.1777112        total: 2.21s    remaining: 336ms
868:    learn: 0.1775958        total: 2.21s    remaining: 333ms
869:    learn: 0.1775294        total: 2.21s    remaining: 331ms
870:    learn: 0.1774664        total: 2.21s    remaining: 328ms
871:    learn: 0.1773815        total: 2.22s    remaining: 326ms
872:    learn: 0.1773007        total: 2.22s    remaining: 323ms
873:    learn: 0.1772166        total: 2.22s    remaining: 321ms
874:    learn: 0.1770894        total: 2.23s    remaining: 318ms
875:    learn: 0.1770106        total: 2.23s    remaining: 316ms
876:    learn: 0.1769308        total: 2.23s    remaining: 313ms
877:    learn: 0.1768260        total: 2.23s    remaining: 311ms
878:    learn: 0.1767294        total: 2.24s    remaining: 308ms
879:    learn: 0.1766212        total: 2.24s    remaining: 306ms
880:    learn: 0.1765554        total: 2.25s    remaining: 303ms
881:    learn: 0.1764990        total: 2.25s    remaining: 301ms
882:    learn: 0.1764196        total: 2.25s    remaining: 298ms
883:    learn: 0.1762940        total: 2.25s    remaining: 296ms
884:    learn: 0.1762324        total: 2.26s    remaining: 293ms
885:    learn: 0.1761649        total: 2.26s    remaining: 291ms
886:    learn: 0.1760673        total: 2.26s    remaining: 288ms
887:    learn: 0.1759986        total: 2.27s    remaining: 286ms
888:    learn: 0.1759168        total: 2.27s    remaining: 283ms
889:    learn: 0.1758304        total: 2.27s    remaining: 281ms
```

```
890:    learn: 0.1757425        total: 2.27s    remaining: 278ms
891:    learn: 0.1756735        total: 2.27s    remaining: 276ms
892:    learn: 0.1755741        total: 2.28s    remaining: 273ms
893:    learn: 0.1754783        total: 2.28s    remaining: 270ms
894:    learn: 0.1753533        total: 2.28s    remaining: 268ms
895:    learn: 0.1752493        total: 2.29s    remaining: 265ms
896:    learn: 0.1751236        total: 2.29s    remaining: 263ms
897:    learn: 0.1750159        total: 2.29s    remaining: 260ms
898:    learn: 0.1749316        total: 2.29s    remaining: 258ms
899:    learn: 0.1747999        total: 2.29s    remaining: 255ms
900:    learn: 0.1746780        total: 2.3s     remaining: 253ms
901:    learn: 0.1745496        total: 2.3s     remaining: 250ms
902:    learn: 0.1744692        total: 2.3s     remaining: 247ms
903:    learn: 0.1742881        total: 2.31s    remaining: 245ms
904:    learn: 0.1742116        total: 2.31s    remaining: 242ms
905:    learn: 0.1741047        total: 2.31s    remaining: 240ms
906:    learn: 0.1740323        total: 2.31s    remaining: 237ms
907:    learn: 0.1739681        total: 2.31s    remaining: 235ms
908:    learn: 0.1739005        total: 2.32s    remaining: 232ms
909:    learn: 0.1738190        total: 2.32s    remaining: 229ms
910:    learn: 0.1737295        total: 2.32s    remaining: 227ms
911:    learn: 0.1736260        total: 2.32s    remaining: 224ms
912:    learn: 0.1735277        total: 2.33s    remaining: 222ms
913:    learn: 0.1734357        total: 2.33s    remaining: 219ms
914:    learn: 0.1733909        total: 2.33s    remaining: 217ms
915:    learn: 0.1732814        total: 2.33s    remaining: 214ms
916:    learn: 0.1731848        total: 2.34s    remaining: 211ms
917:    learn: 0.1730713        total: 2.34s    remaining: 209ms
918:    learn: 0.1730084        total: 2.34s    remaining: 206ms
919:    learn: 0.1729167        total: 2.34s    remaining: 204ms
920:    learn: 0.1728425        total: 2.35s    remaining: 201ms
921:    learn: 0.1727290        total: 2.35s    remaining: 199ms
922:    learn: 0.1725846        total: 2.35s    remaining: 196ms
923:    learn: 0.1724833        total: 2.35s    remaining: 194ms
924:    learn: 0.1724279        total: 2.35s    remaining: 191ms
925:    learn: 0.1722385        total: 2.36s    remaining: 188ms
926:    learn: 0.1721559        total: 2.36s    remaining: 186ms
927:    learn: 0.1720461        total: 2.36s    remaining: 183ms
928:    learn: 0.1719993        total: 2.36s    remaining: 181ms
929:    learn: 0.1719384        total: 2.37s    remaining: 178ms
930:    learn: 0.1719112        total: 2.37s    remaining: 176ms
931:    learn: 0.1718294        total: 2.37s    remaining: 173ms
932:    learn: 0.1717948        total: 2.37s    remaining: 170ms
933:    learn: 0.1717062        total: 2.38s    remaining: 168ms
934:    learn: 0.1716402        total: 2.38s    remaining: 165ms
935:    learn: 0.1714960        total: 2.38s    remaining: 163ms
936:    learn: 0.1714460        total: 2.38s    remaining: 160ms
937:    learn: 0.1713551        total: 2.38s    remaining: 158ms
```

```
938:    learn: 0.1713095        total: 2.39s    remaining: 155ms
939:    learn: 0.1712494        total: 2.39s    remaining: 153ms
940:    learn: 0.1711883        total: 2.39s    remaining: 150ms
941:    learn: 0.1710755        total: 2.4s     remaining: 148ms
942:    learn: 0.1709994        total: 2.4s     remaining: 145ms
943:    learn: 0.1708740        total: 2.4s     remaining: 142ms
944:    learn: 0.1708236        total: 2.4s     remaining: 140ms
945:    learn: 0.1707626        total: 2.4s     remaining: 137ms
946:    learn: 0.1706987        total: 2.41s    remaining: 135ms
947:    learn: 0.1705929        total: 2.41s    remaining: 132ms
948:    learn: 0.1704582        total: 2.41s    remaining: 130ms
949:    learn: 0.1703705        total: 2.41s    remaining: 127ms
950:    learn: 0.1702928        total: 2.42s    remaining: 125ms
951:    learn: 0.1702014        total: 2.42s    remaining: 122ms
952:    learn: 0.1701227        total: 2.42s    remaining: 119ms
953:    learn: 0.1700540        total: 2.42s    remaining: 117ms
954:    learn: 0.1699134        total: 2.42s    remaining: 114ms
955:    learn: 0.1698320        total: 2.43s    remaining: 112ms
956:    learn: 0.1697319        total: 2.43s    remaining: 109ms
957:    learn: 0.1696273        total: 2.43s    remaining: 107ms
958:    learn: 0.1695254        total: 2.44s    remaining: 104ms
959:    learn: 0.1694307        total: 2.44s    remaining: 102ms
960:    learn: 0.1693773        total: 2.44s    remaining: 99.1ms
961:    learn: 0.1692911        total: 2.44s    remaining: 96.6ms
962:    learn: 0.1692065        total: 2.45s    remaining: 94ms
963:    learn: 0.1691139        total: 2.45s    remaining: 91.5ms
964:    learn: 0.1690015        total: 2.45s    remaining: 88.9ms
965:    learn: 0.1688710        total: 2.45s    remaining: 86.4ms
966:    learn: 0.1688199        total: 2.46s    remaining: 83.9ms
967:    learn: 0.1686431        total: 2.46s    remaining: 81.3ms
968:    learn: 0.1685472        total: 2.46s    remaining: 78.8ms
969:    learn: 0.1684542        total: 2.46s    remaining: 76.2ms
970:    learn: 0.1683823        total: 2.47s    remaining: 73.7ms
971:    learn: 0.1682858        total: 2.47s    remaining: 71.2ms
972:    learn: 0.1681957        total: 2.47s    remaining: 68.6ms
973:    learn: 0.1680997        total: 2.47s    remaining: 66.1ms
974:    learn: 0.1679852        total: 2.48s    remaining: 63.5ms
975:    learn: 0.1679046        total: 2.48s    remaining: 61ms
976:    learn: 0.1678101        total: 2.48s    remaining: 58.5ms
977:    learn: 0.1677251        total: 2.48s    remaining: 55.9ms
978:    learn: 0.1676415        total: 2.49s    remaining: 53.4ms
979:    learn: 0.1675496        total: 2.49s    remaining: 50.8ms
980:    learn: 0.1674598        total: 2.49s    remaining: 48.3ms
981:    learn: 0.1673210        total: 2.5s     remaining: 45.7ms
982:    learn: 0.1672652        total: 2.5s     remaining: 43.2ms
983:    learn: 0.1671754        total: 2.5s     remaining: 40.7ms
984:    learn: 0.1671456        total: 2.5s     remaining: 38.1ms
985:    learn: 0.1670289        total: 2.51s    remaining: 35.6ms
```

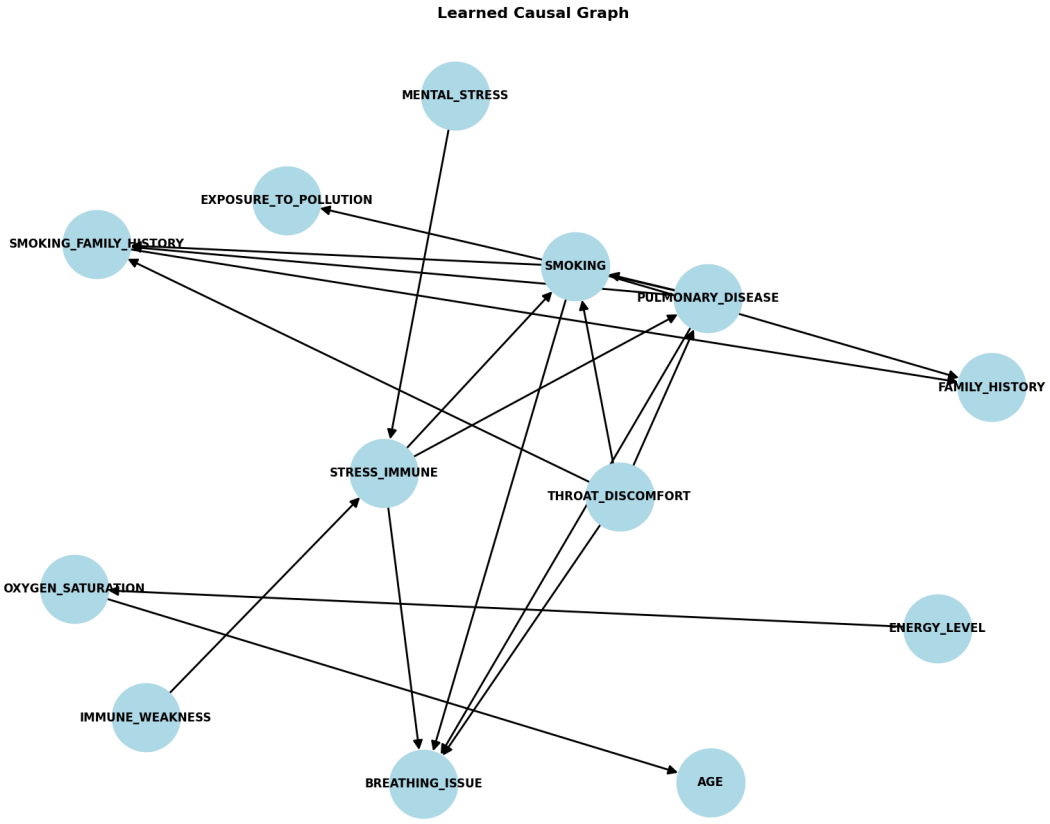
```
986:    learn: 0.1669388        total: 2.51s    remaining: 33ms
987:    learn: 0.1668287        total: 2.51s    remaining: 30.5ms
988:    learn: 0.1667586        total: 2.51s    remaining: 28ms
989:    learn: 0.1666944        total: 2.52s    remaining: 25.4ms
990:    learn: 0.1666007        total: 2.52s    remaining: 22.9ms
991:    learn: 0.1665093        total: 2.52s    remaining: 20.3ms
992:    learn: 0.1664415        total: 2.52s    remaining: 17.8ms
993:    learn: 0.1663632        total: 2.53s    remaining: 15.2ms
994:    learn: 0.1662863        total: 2.53s    remaining: 12.7ms
995:    learn: 0.1662254        total: 2.53s    remaining: 10.2ms
996:    learn: 0.1661765        total: 2.53s    remaining: 7.62ms
997:    learn: 0.1661025        total: 2.54s    remaining: 5.08ms
998:    learn: 0.1660400        total: 2.54s    remaining: 2.54ms
999:    learn: 0.1659847        total: 2.54s    remaining: 0us
CAT: AUC=0.917, F1=0.876
SVM: AUC=0.817, F1=0.000
KNN: AUC=0.664, F1=0.548
Naive Bayes: AUC=0.884, F1=0.826
110/110             1s 1ms/step -
accuracy: 0.4372 - loss: 10.5873
47/47             0s 2ms/step
47/47             0s 2ms/step
SEQ DENSE: AUC=0.507, F1=0.112
110/110             2s 2ms/step -
accuracy: 0.4917 - loss: 6.8923
47/47             0s 2ms/step
47/47             0s 1ms/step
SEQ DROPOUT: AUC=0.590, F1=0.000
```

Causual Inference:

[102]:
```python
import pandas as pd
from pgmpy.estimators import HillClimbSearch
from pgmpy.models.BayesianNetwork import BayesianNetwork

import networkx as nx
import matplotlib.pyplot as plt
from pgmpy.estimators import BDeu



# Load Data
data = data_cleaned  # Replace with actual dataset

# Select relevant features
features = [
    "AGE", "GENDER", "SMOKING", "FINGER_DISCOLORATION", "MENTAL_STRESS",
    "EXPOSURE_TO_POLLUTION", "LONG_TERM_ILLNESS", "ENERGY_LEVEL",
```

```
    "IMMUNE_WEAKNESS", "BREATHING_ISSUE", "ALCOHOL_CONSUMPTION",
    "THROAT_DISCOMFORT", "OXYGEN_SATURATION", "CHEST_TIGHTNESS",
    "FAMILY_HISTORY", "SMOKING_FAMILY_HISTORY", "STRESS_IMMUNE",
    "PULMONARY_DISEASE"
]

data = data[features]  # Filter required features

# Learn Causal Structure using Hill Climbing

hc = HillClimbSearch(data)
best_model = hc.estimate(scoring_method=BDeu(data),max_iter=20)


# Convert to NetworkX graph
G = nx.DiGraph(best_model.edges())

# Plot Graph
plt.figure(figsize=(16, 12))
pos = nx.spring_layout(G, k=2)
nx.draw(G, pos, with_labels=True, node_color="lightblue", edge_color="black",
        node_size=5000, font_size=12, font_weight="bold", arrowsize=20, width=2)
plt.title("Learned Causal Graph", fontsize=16, fontweight="bold")
plt.show()
```

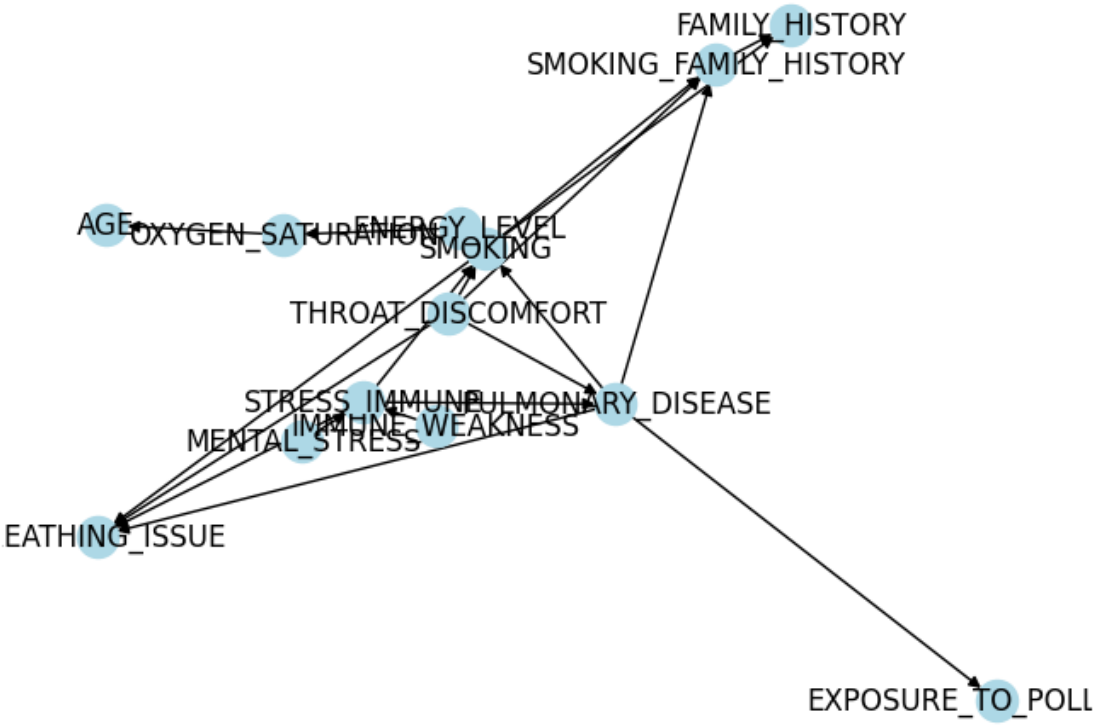
```
  0%|          | 0/20 [00:00<?, ?it/s]
```



[104]:
```
import pgmpy.estimators
print(dir(pgmpy.estimators))
```

```
['AIC', 'AICCondGauss', 'AICGauss', 'BDeu', 'BDs', 'BIC', 'BICCondGauss',
'BICGauss', 'BaseEstimator', 'BayesianEstimator', 'CITests', 'EM',
'ExhaustiveSearch', 'ExpectationMaximization', 'ExpertInLoop',
'ExpertKnowledge', 'GES', 'HillClimbSearch', 'IVEstimator', 'K2', 'LinearModel',
'LogLikelihoodCondGauss', 'LogLikelihoodGauss', 'MLE', 'MarginalEstimator',
'MaximumLikelihoodEstimator', 'MirrorDescentEstimator', 'MmhcEstimator', 'PC',
'ParameterEstimator', 'SEMEstimator', 'ScoreCache', 'StructureEstimator',
'StructureScore', 'TreeSearch', '__all__', '__builtins__', '__cached__',
'__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__',
'__spec__', 'base', 'expert', 'get_scoring_method']
```

[120]:
```
import networkx as nx
import matplotlib.pyplot as plt

nx.draw(G, with_labels=True, node_color="lightblue", edge_color="black")
plt.show()
```



[123]:
```
from networkx.algorithms.cycles import find_cycle

try:
    cycle = find_cycle(best_model)
    print("Cycle detected:", cycle)
except:
    print("No cycles found, the graph is a valid DAG")
```

```
No cycles found, the graph is a valid DAG
```

[124]:
```
from pgmpy.estimators import PC

pc_estimator = PC(data)
dag = pc_estimator.estimate()
dag.edges()  # Check learned causal structure
```

```
  0%|          | 0/5 [00:00<?, ?it/s]
```

[124]:
```
OutEdgeView([('MENTAL_STRESS', 'STRESS_IMMUNE'), ('STRESS_IMMUNE',
'PULMONARY_DISEASE'), ('IMMUNE_WEAKNESS', 'STRESS_IMMUNE'),
('EXPOSURE_TO_POLLUTION', 'PULMONARY_DISEASE'), ('SMOKING',
```

```
'PULMONARY_DISEASE'), ('SMOKING', 'SMOKING_FAMILY_HISTORY'), ('BREATHING_ISSUE',
'PULMONARY_DISEASE'), ('SMOKING_FAMILY_HISTORY', 'PULMONARY_DISEASE'),
('THROAT_DISCOMFORT', 'PULMONARY_DISEASE'), ('FAMILY_HISTORY',
'SMOKING_FAMILY_HISTORY')])
```

[125]:
```
from pgmpy.models import DiscreteBayesianNetwork

# Convert your DAG to a Bayesian Network
bayesian_model = DiscreteBayesianNetwork(best_model.edges())

# Now, apply MLE
from pgmpy.estimators import MaximumLikelihoodEstimator

mle_estimator = MaximumLikelihoodEstimator(bayesian_model, data)

# for node in bayesian_model.nodes():
  #  print(mle_estimator.estimate_cpd(node))
```

[126]:
```
from pgmpy.inference import VariableElimination

# Attach CPDs to the model
bayesian_model.fit(data, estimator=MaximumLikelihoodEstimator)

# Verify CPDs
# for cpd in bayesian_model.get_cpds():
 #   print(cpd)
```

[126]:
```
<pgmpy.models.DiscreteBayesianNetwork.DiscreteBayesianNetwork at 0x7bb7f04c8b20>
```

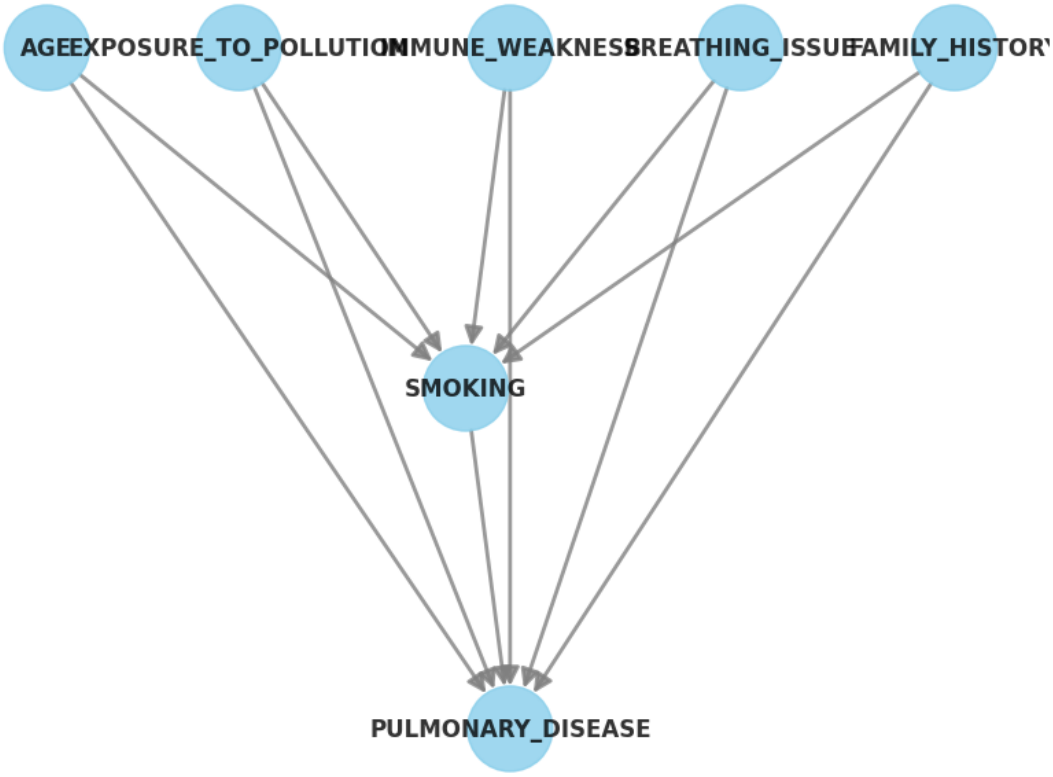
[127]:
```
inference = VariableElimination(bayesian_model)

# Example Query: P(Pulmonary_Disease | Smoking = 1)
result = inference.query(variables=['PULMONARY_DISEASE'], evidence={'SMOKING':␣
 ↪1})
print(result)
```

```
+----------------------+--------------------------+
| PULMONARY_DISEASE    |   phi(PULMONARY_DISEASE) |
+======================+==========================+
| PULMONARY_DISEASE(0) |                   0.4313 |
+----------------------+--------------------------+
| PULMONARY_DISEASE(1) |                   0.5687 |
+----------------------+--------------------------+
```

!pip install dowhy

[128]:
```
  # Install DoWhy if not already installed
import dowhy
```

```
from dowhy import CausalModel
import matplotlib.pyplot as plt

model = CausalModel(
    data=data,  # Your dataset
    treatment="SMOKING",  # The cause (X)
    outcome="PULMONARY_DISEASE",  # The effect (Y)
    common_causes=["AGE","EXPOSURE_TO_POLLUTION","IMMUNE_WEAKNESS",␣
 ↪"BREATHING_ISSUE","FAMILY_HISTORY"]  # Confounders
)

model.view_model(layout="dot")  # Correct method to visualize causal graph
plt.show()
```



[129]:
```
identified_estimand = model.identify_effect()
print(identified_estimand)
```

```
Estimand type: EstimandType.NONPARAMETRIC_ATE

### Estimand : 1
Estimand name: backdoor
```

```
Estimand expression:
    d

     (E[PULMONARY_DISEASE|BREATHING_ISSUE,FAMILY_HISTORY,EXPOSURE_TO_POLLUT
ION,IMMUNE_WEAKNES
d[SMOKING]



 S,AGE])

Estimand assumption 1, Unconfoundedness: If U→{SMOKING} and U→PULMONARY_DISEASE
then P(PULMONARY_DISEASE|SMOKING,BREATHING_ISSUE,FAMILY_HISTORY,EXPOSURE_TO_POLL
UTION,IMMUNE_WEAKNESS,AGE,U) = P(PULMONARY_DISEASE|SMOKING,BREATHING_ISSUE,FAMIL
Y_HISTORY,EXPOSURE_TO_POLLUTION,IMMUNE_WEAKNESS,AGE)

### Estimand : 2
Estimand name: iv
No such variable(s) found!

### Estimand : 3
Estimand name: frontdoor
No such variable(s) found!
```

[134]:
```
estimate = model.estimate_effect(identified_estimand, method_name="backdoor.
 ↪propensity_score_matching")
print(estimate)
```

```
*** Causal Estimate ***

## Identified estimand
Estimand type: EstimandType.NONPARAMETRIC_ATE

### Estimand : 1
Estimand name: backdoor
Estimand expression:
    d

     (E[PULMONARY_DISEASE|BREATHING_ISSUE,FAMILY_HISTORY,EXPOSURE_TO_POLLUT
ION,IMMUNE_WEAKNES
d[SMOKING]



 S,AGE])

Estimand assumption 1, Unconfoundedness: If U→{SMOKING} and U→PULMONARY_DISEASE
```

```
then P(PULMONARY_DISEASE|SMOKING,BREATHING_ISSUE,FAMILY_HISTORY,EXPOSURE_TO_POLL
UTION,IMMUNE_WEAKNESS,AGE,U) = P(PULMONARY_DISEASE|SMOKING,BREATHING_ISSUE,FAMIL
Y_HISTORY,EXPOSURE_TO_POLLUTION,IMMUNE_WEAKNESS,AGE)

## Realized estimand
b: PULMONARY_DISEASE~SMOKING+BREATHING_ISSUE+FAMILY_HISTORY+EXPOSURE_TO_POLLUTIO
N+IMMUNE_WEAKNESS+AGE
Target units: ate

## Estimate
Mean value: 0.4816
```

[139]:
```
# Generate counterfactual data
counterfactuals = model.refute_estimate(identified_estimand, estimate,␣
  ↪method_name="placebo_treatment_refuter",num_simulations=100)
print(counterfactuals)
```

```
Refute: Use a Placebo Treatment
Estimated effect:0.4816
New effect:-0.0015100000000000005
p value:0.98
```

[140]:
```
print(counterfactuals.__dict__)  # Print all attributes of the object
```

```
{'estimated_effect': 0.4816, 'new_effect': -0.0015100000000000005,
'refutation_type': 'Refute: Use a Placebo Treatment', 'refutation_result':
{'p_value': 0.98, 'is_statistically_significant': False}, 'refuter':
<dowhy.causal_refuters.placebo_treatment_refuter.PlaceboTreatmentRefuter object
at 0x7bb7f06c9ed0>}
```

[136]:
```
sensitivity_analysis = model.refute_estimate(identified_estimand, estimate,␣
  ↪method_name="random_common_cause")
print(sensitivity_analysis)
```

```
Refute: Add a random common cause
Estimated effect:0.4816
New effect:0.4816
p value:1.0
```