```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

```
df=pd.read_excel('/content/ENB2012_data[1].xlsx')
```

```
df
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0.98	514.5	294.0	110.25	7.0	2	0.0	0	15.55	21.33
1	0.98	514.5	294.0	110.25	7.0	3	0.0	0	15.55	21.33
2	0.98	514.5	294.0	110.25	7.0	4	0.0	0	15.55	21.33
3	0.98	514.5	294.0	110.25	7.0	5	0.0	0	15.55	21.33
4	0.90	563.5	318.5	122.50	7.0	2	0.0	0	20.84	28.28
...	...	...	...	...	...	...	...	...	...	...
763	0.64	784.0	343.0	220.50	3.5	5	0.4	5	17.88	21.40
764	0.62	808.5	367.5	220.50	3.5	2	0.4	5	16.54	16.88
765	0.62	808.5	367.5	220.50	3.5	3	0.4	5	16.44	17.11
766	0.62	808.5	367.5	220.50	3.5	4	0.4	5	16.48	16.61
767	0.62	808.5	367.5	220.50	3.5	5	0.4	5	16.64	16.03

768 rows × 10 columns

Next steps: Generate code with df New interactive sheet

```
df.shape
```

```
(768, 10)
```

```
df.dtypes
```

	0
X1	float64
X2	float64
X3	float64
X4	float64
X5	float64
X6	int64
X7	float64
X8	int64
Y1	float64
Y2	float64

dtype: object

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   X1      768 non-null    float64
 1   X2      768 non-null    float64
```

```
 2   X3      768 non-null    float64
 3   X4      768 non-null    float64
 4   X5      768 non-null    float64
 5   X6      768 non-null    int64
 6   X7      768 non-null    float64
 7   X8      768 non-null    int64
 8   Y1      768 non-null    float64
 9   Y2      768 non-null    float64
dtypes: float64(8), int64(2)
memory usage: 60.1 KB
```

```
df.describe()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.00000	768.000000	768.000000
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.81250	22.307195	24.587760
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.55096	10.090204	9.513306
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.00000	6.010000	10.900000
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.75000	12.992500	15.620000
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.00000	18.950000	22.080000
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.00000	31.667500	33.132500
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.00000	43.100000	48.030000

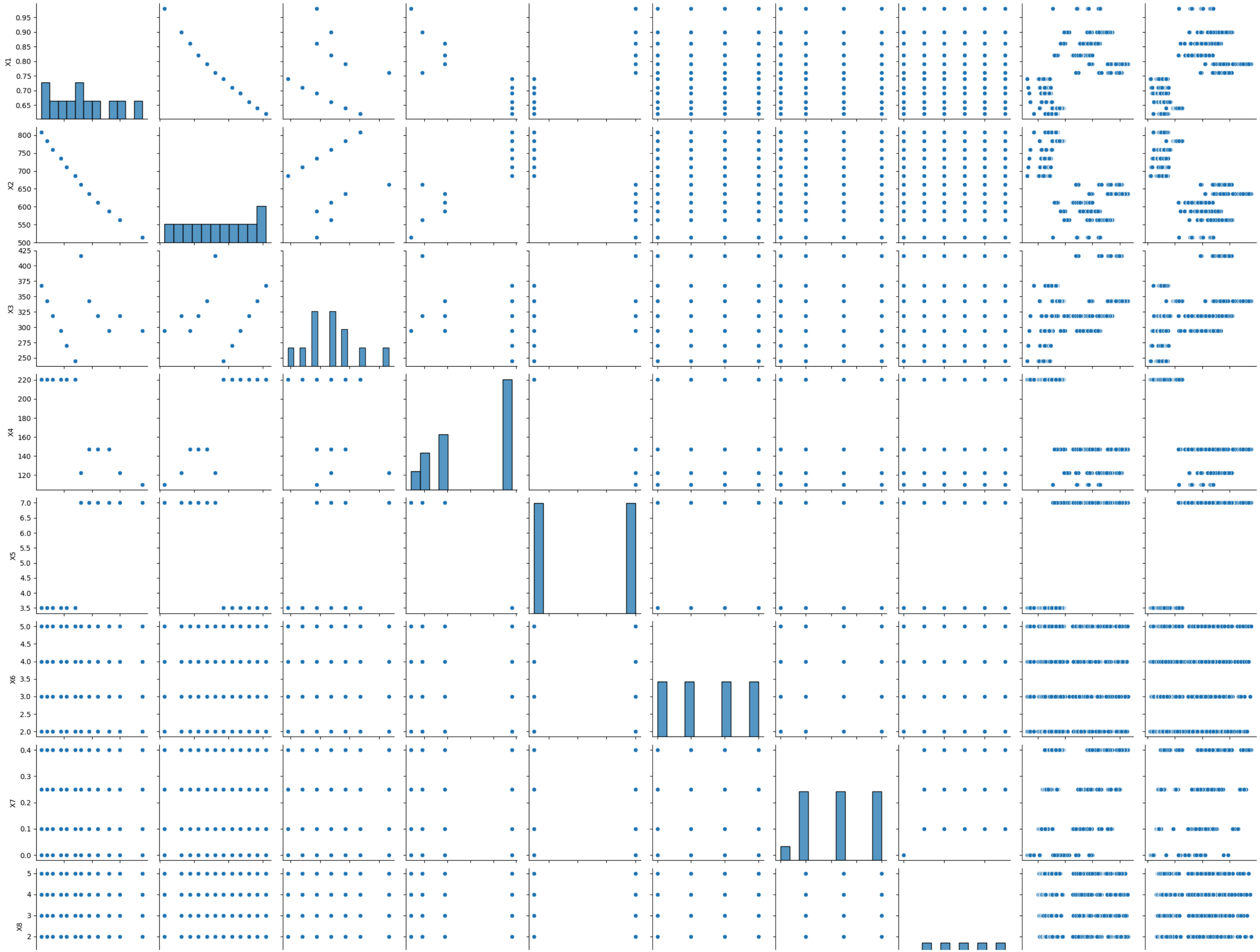
```
df.isnull().sum()
```

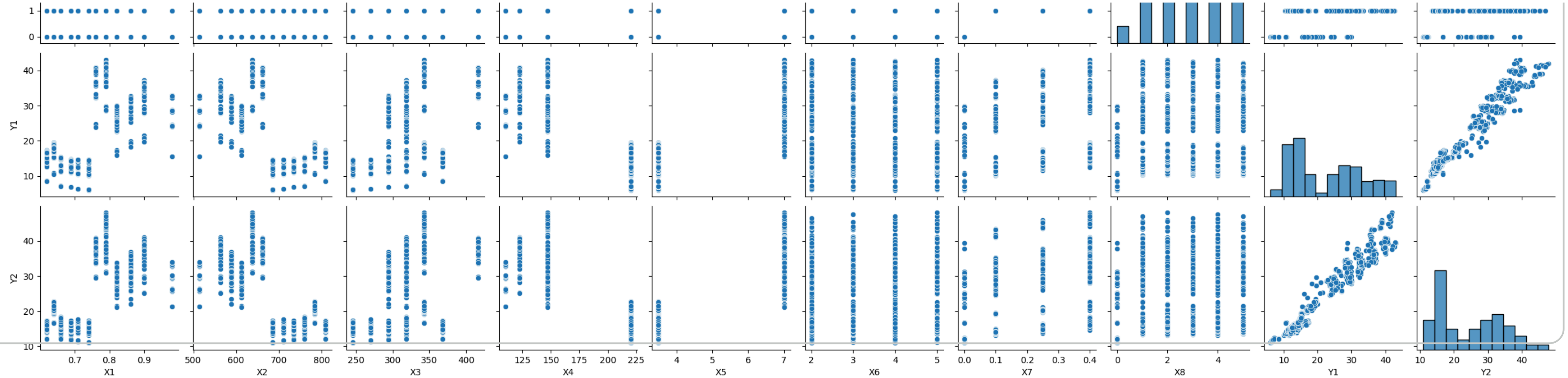
	0
X1	0
X2	0
X3	0
X4	0
X5	0
X6	0
X7	0
X8	0
Y1	0
Y2	0

dtype: int64

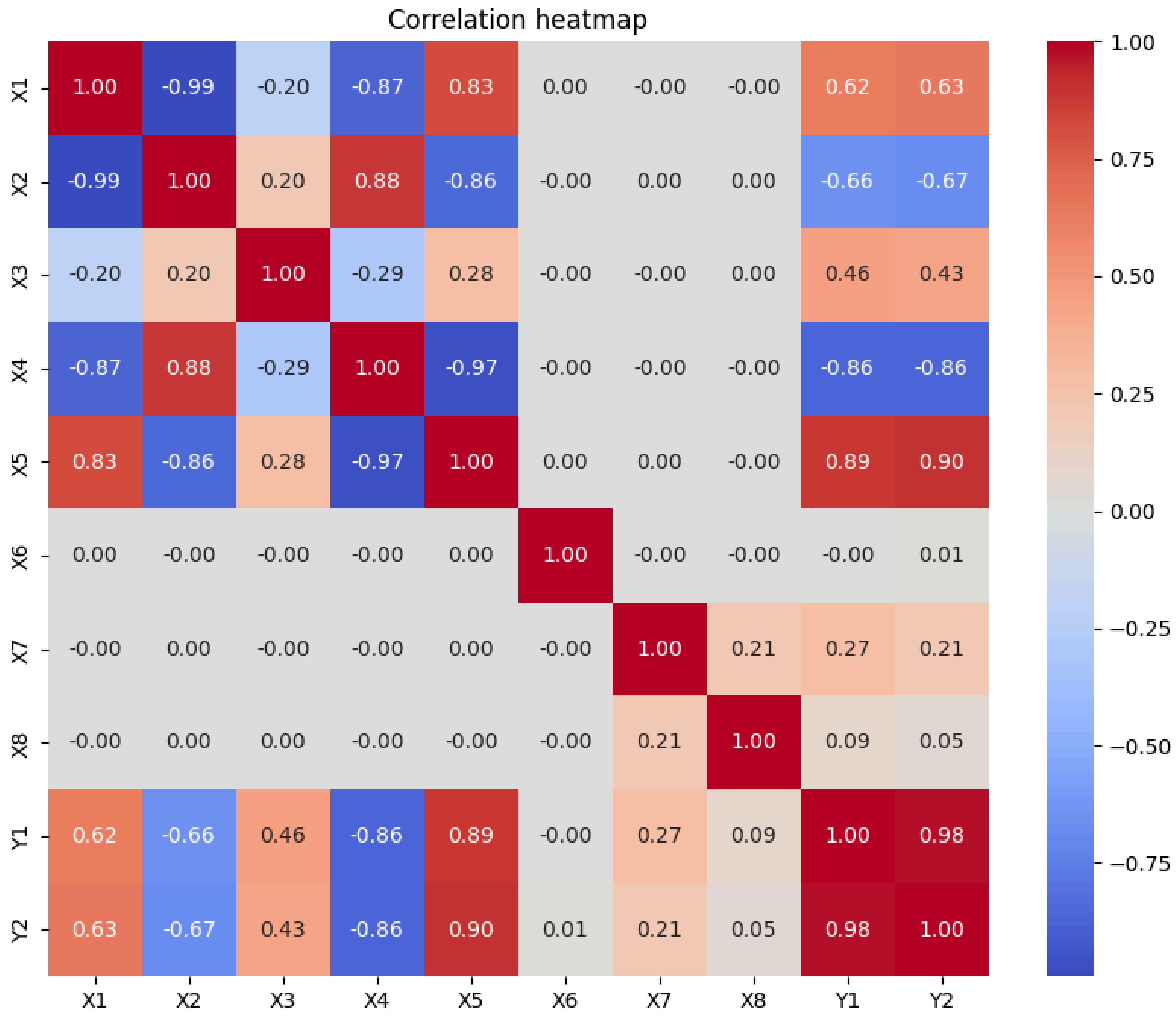
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7c524f849040>
```

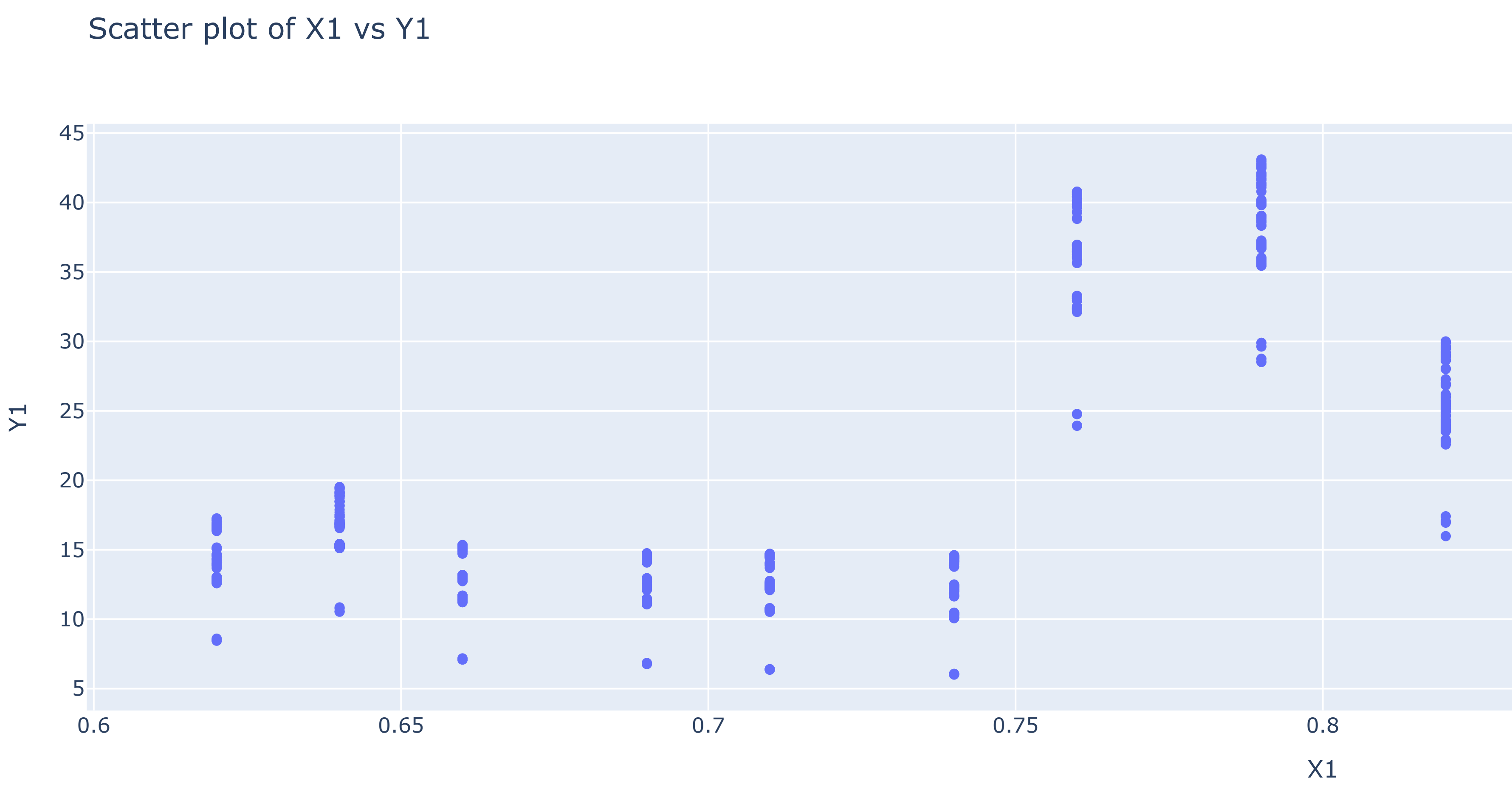



Y1
Y2
X1
X2
X3
X4
X5
X6
X7
X8
Y1
Y2

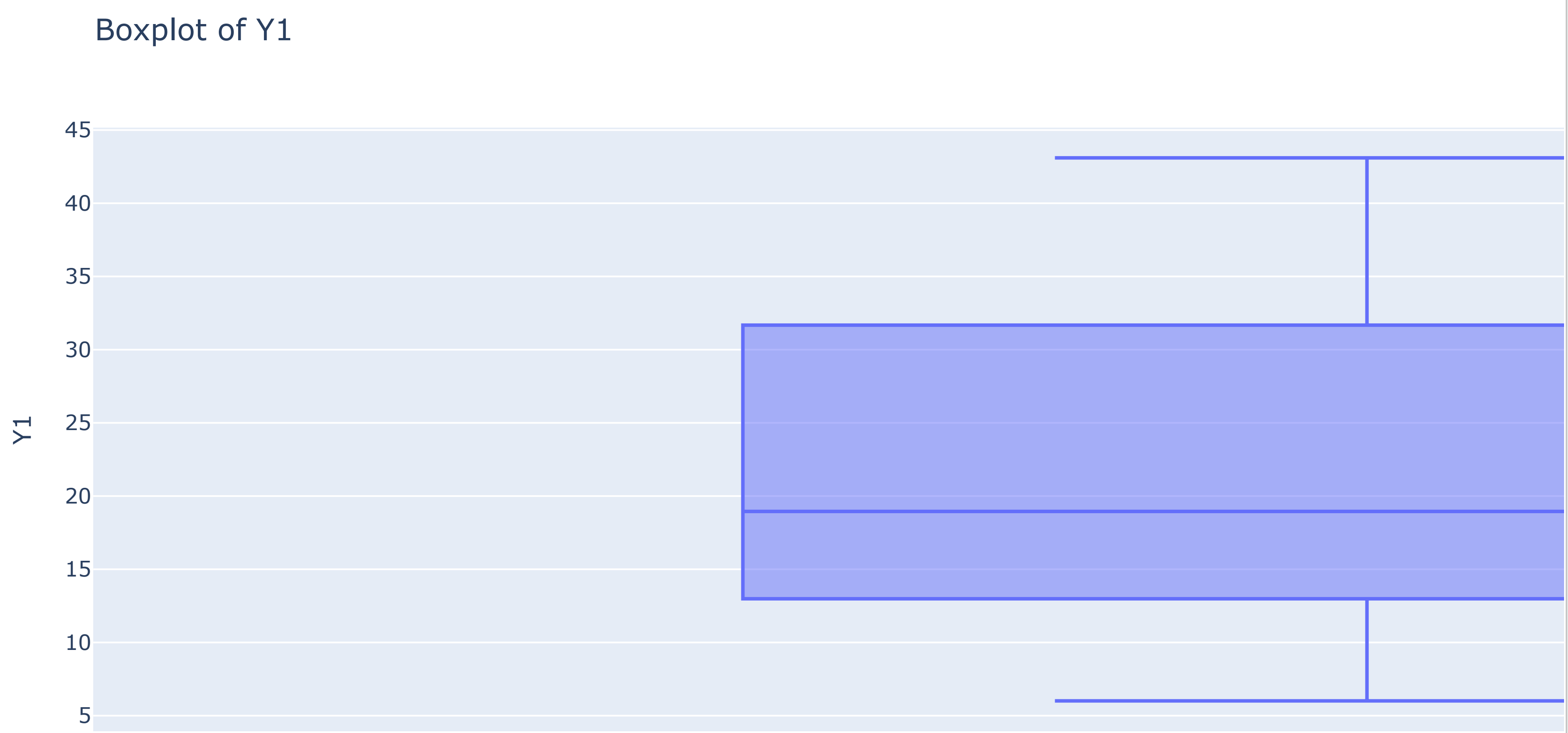
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm',fmt=".2f")
plt.title('Correlation heatmap')
plt.show()
```


Correlation heatmap
X1 X2 X3 X4 X5 X6 X7 X8 Y1 Y2
X1: 1.00 -0.99 -0.20 -0.87 0.83 0.00 -0.00 -0.00 0.62 0.63
X2: -0.99 1.00 0.20 0.88 -0.86 -0.00 0.00 0.00 -0.66 -0.67
X3: -0.20 0.20 1.00 -0.29 0.28 -0.00 -0.00 0.00 0.46 0.43
X4: -0.87 0.88 -0.29 1.00 -0.97 -0.00 -0.00 -0.00 -0.86 -0.86
X5: 0.83 -0.86 0.28 -0.97 1.00 0.00 0.00 -0.00 0.89 0.90
X6: 0.00 -0.00 -0.00 -0.00 0.00 1.00 -0.00 -0.00 -0.00 0.01
X7: -0.00 0.00 -0.00 -0.00 0.00 -0.00 1.00 0.21 0.27 0.21
X8: -0.00 0.00 0.00 -0.00 -0.00 -0.00 0.21 1.00 0.09 0.05
Y1: 0.62 -0.66 0.46 -0.86 0.89 -0.00 0.27 0.09 1.00 0.98
Y2: 0.63 -0.67 0.43 -0.86 0.90 0.01 0.21 0.05 0.98 1.00
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75

```
fig = px.scatter(df, x='X1', y='Y1', title='Scatter plot of X1 vs Y1')
fig.show()
```




```
fig = px.box(df, y='Y1', title='Boxplot of Y1')
fig.show()
```




```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
from sklearn.linear_model import LinearRegression
```

```
X = df.iloc[:, 0:8]
Y1 = df['Y1']
Y2 = df['Y2']
```

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, Y1, test_size=0.2, random_state=42)
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, Y2, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train1)
X_test_scaled = scaler.transform(X_test1)
```

```
lr = LinearRegression()
# Train model for Heating Load (Y1)
lr.fit(X_train1, y_train1)
y_pred1 = lr.predict(X_test1)
```

```
mae1 = mean_absolute_error(y_test1, y_pred1)
rmse1 = np.sqrt(mean_squared_error(y_test1, y_pred1))
r2_1 = r2_score(y_test1, y_pred1)
```

```
#evaluation for Y1
print(f"MAE  : {mae1:.3f}")
print(f"RMSE : {rmse1:.3f}")
print(f"R²   : {r2_1:.3f}")
```

```
MAE  : 2.182
RMSE : 3.025
R²   : 0.912
```

```
# Train model for Cooling Load (Y2)
lr.fit(X_train2, y_train2)
```

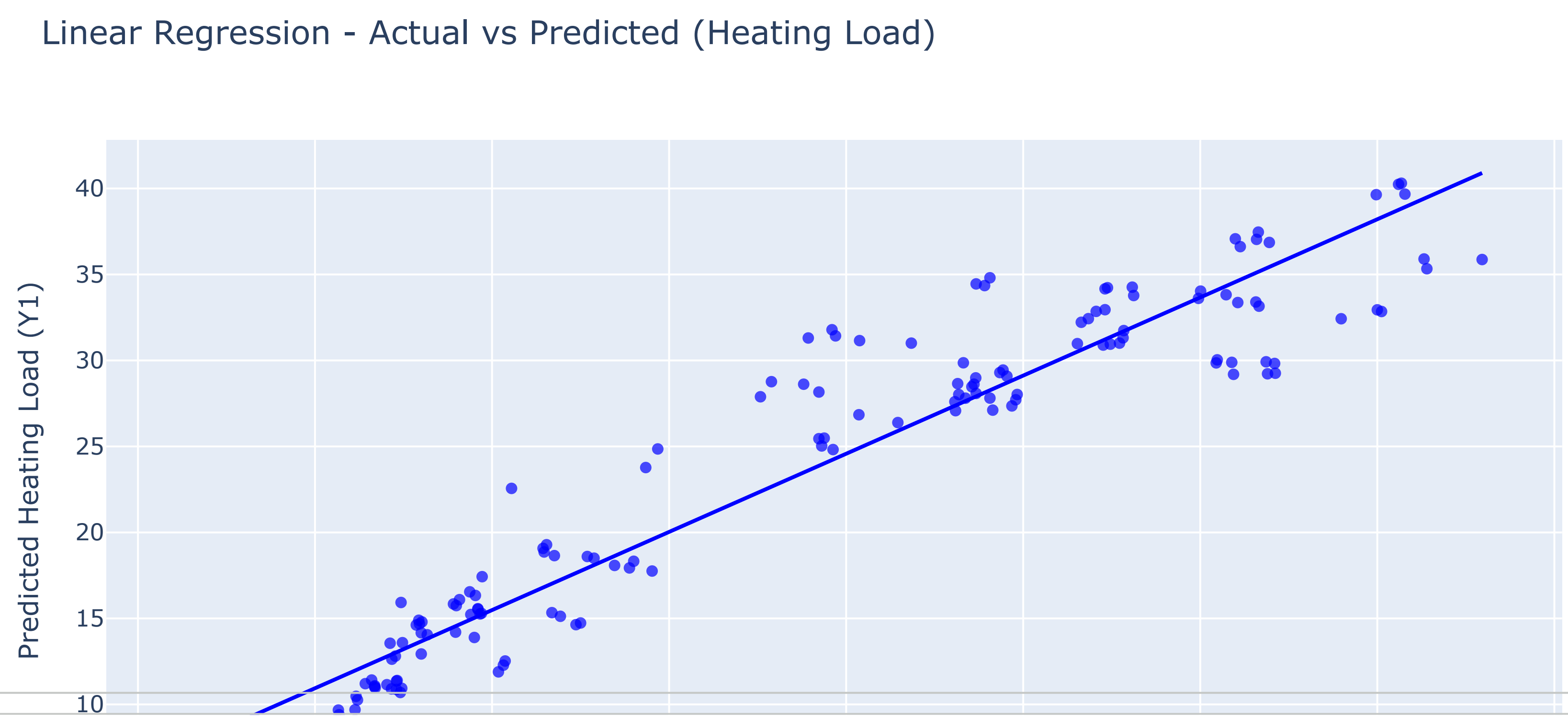
```
y_pred2 = lr.predict(X_test2)
```

```
mae2 = mean_absolute_error(y_test2, y_pred2)
rmse2 = np.sqrt(mean_squared_error(y_test2, y_pred2))
r2_2 = r2_score(y_test2, y_pred2)
```

```
print(f"MAE  : {mae2:.3f}")
print(f"RMSE : {rmse2:.3f}")
print(f"R²   : {r2_2:.3f}")
```

```
MAE  : 2.195
RMSE : 3.145
R²   : 0.893
```

```
# Heating Load (Y1)
fig1 = px.scatter(
    x=y_test1, y=y_pred1,
    labels={'x': 'Actual Heating Load (Y1)', 'y': 'Predicted Heating Load (Y1)'},
    title='Linear Regression - Actual vs Predicted (Heating Load)',
    trendline='ols'
)
fig1.update_traces(marker=dict(size=6, color='blue', opacity=0.7))
fig1.show()
```




```
# Cooling Load (Y2)
fig2 = px.scatter(
```