In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Classification

In [2]:
```
df_class = pd.read_csv("../data/classification.csv")
```

In [3]:
```
df_class.shape
```

Out[3]:
```
(1500, 16)
```

In [4]:
```
df_class.dtypes
```

Out[4]:
```
X1     float64
X2     float64
X3     float64
X4     float64
X5     float64
X6     float64
X7     float64
X8     float64
X9     float64
X10    float64
X11    float64
X12    float64
X13    float64
X14    float64
X15    float64
Y        int64
dtype: object
```

In [5]:
```
df_class.head()
```

Out[5]:

	X1	X2	X3	X4	X5	X6	X7	
0	-2.388741	6.221087	3.442447	1.273807	0.912272	8.908027	8.441999	-2.6
1	-6.012792	-9.884413	-1.590610	4.999943	0.247758	-1.197048	-10.939272	1.5
2	2.270829	-8.849332	-6.619179	-2.861520	-6.720253	5.715418	6.493857	-4.4
3	-7.092421	-10.254081	-0.907321	3.712683	-0.567676	0.254027	-10.135377	-0.4
4	-2.246293	7.617936	3.580218	2.412760	3.881735	8.096439	8.372886	-4.6

In [6]:
```
df_class.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 16 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X1      1500 non-null   float64
 1   X2      1497 non-null   float64
 2   X3      1500 non-null   float64
 3   X4      1497 non-null   float64
 4   X5      1499 non-null   float64
 5   X6      1500 non-null   float64
 6   X7      1499 non-null   float64
 7   X8      1499 non-null   float64
 8   X9      1500 non-null   float64
 9   X10     1498 non-null   float64
 10  X11     1498 non-null   float64
 11  X12     1499 non-null   float64
 12  X13     1499 non-null   float64
 13  X14     1500 non-null   float64
 14  X15     1500 non-null   float64
 15  Y       1500 non-null   int64  
dtypes: float64(15), int64(1)
memory usage: 187.6 KB
```

In [7]:
```
df_class.describe()
```

Out[7]:

	X1	X2	X3	X4	X5	X
count	1500.000000	1497.000000	1500.000000	1497.000000	1499.000000	1500.00000
mean	-1.883910	-4.075099	-1.382158	0.920043	-2.471415	4.63442
std	3.834396	7.897076	5.051711	3.539492	3.470882	3.95587
min	-9.028149	-13.311122	-10.556390	-6.713349	-10.050322	-3.46119
25%	-5.695571	-9.919480	-7.157683	-2.919740	-6.301859	0.37110
50%	-1.986221	-8.899808	-0.788744	2.032352	-1.602947	5.12905
75%	1.998095	6.259801	3.629647	3.778093	0.507027	8.33460
max	6.040819	11.162159	7.478247	7.514823	3.915434	12.01723

In [8]:
```
missing_summary = df_class.isnull().sum().to_frame(name="n_missing")
missing_summary["pct_missing"] = missing_summary["n_missing"] / len(df_class
print("\nResumen de valores faltantes:")
missing_summary
```

```
Resumen de valores faltantes:
```

Out[8]:

	n_missing	pct_missing
X1	0	0.000000
X2	3	0.200000
X3	0	0.000000
X4	3	0.200000
X5	1	0.066667
X6	0	0.000000
X7	1	0.066667
X8	1	0.066667
X9	0	0.000000
X10	2	0.133333
X11	2	0.133333
X12	1	0.066667
X13	1	0.066667
X14	0	0.000000
X15	0	0.000000
Y	0	0.000000

In [9]:
```
df_class.shape
```

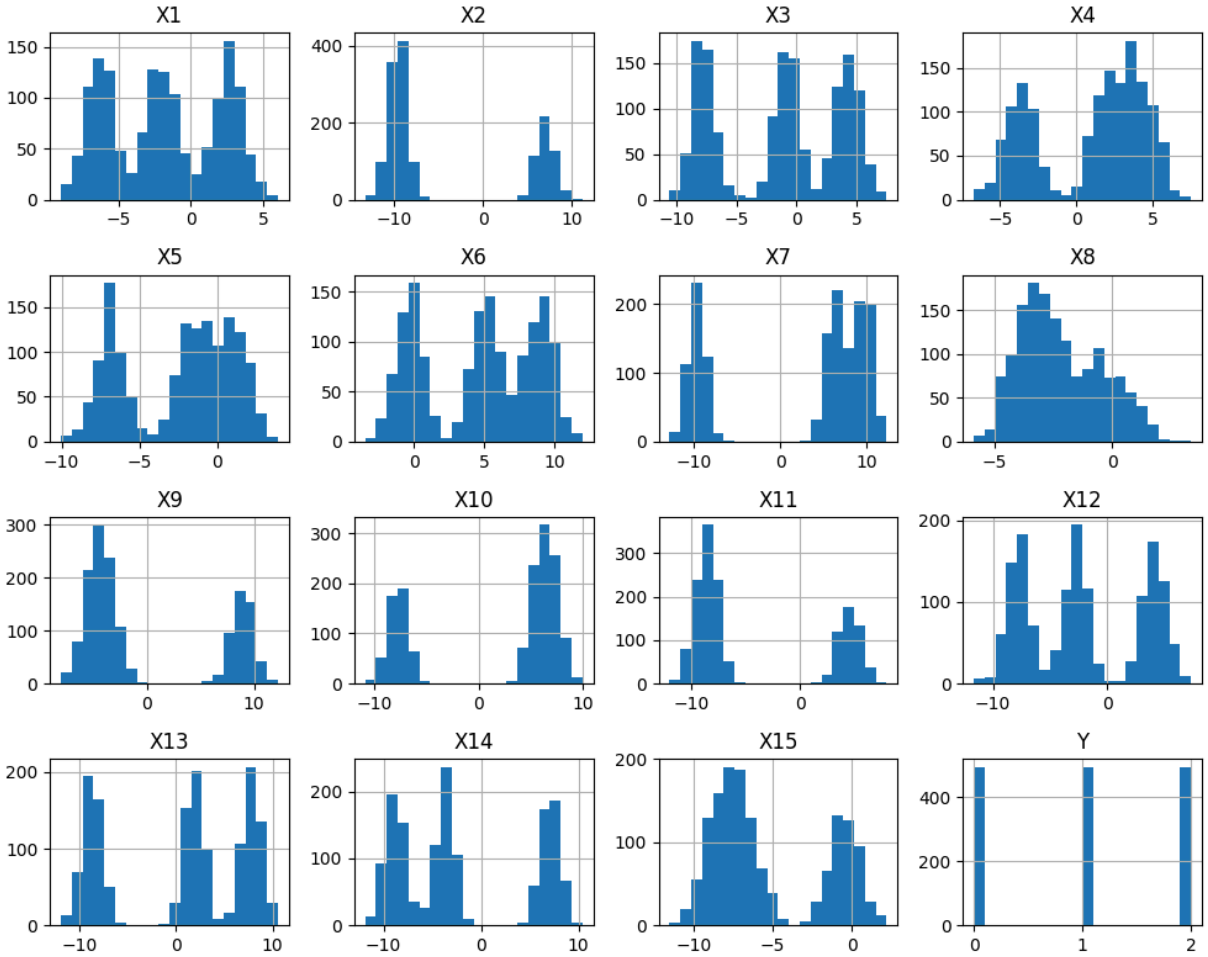
Out[9]:
```
(1500, 16)
```

In [10]:
```
df_class = df_class.dropna()
```

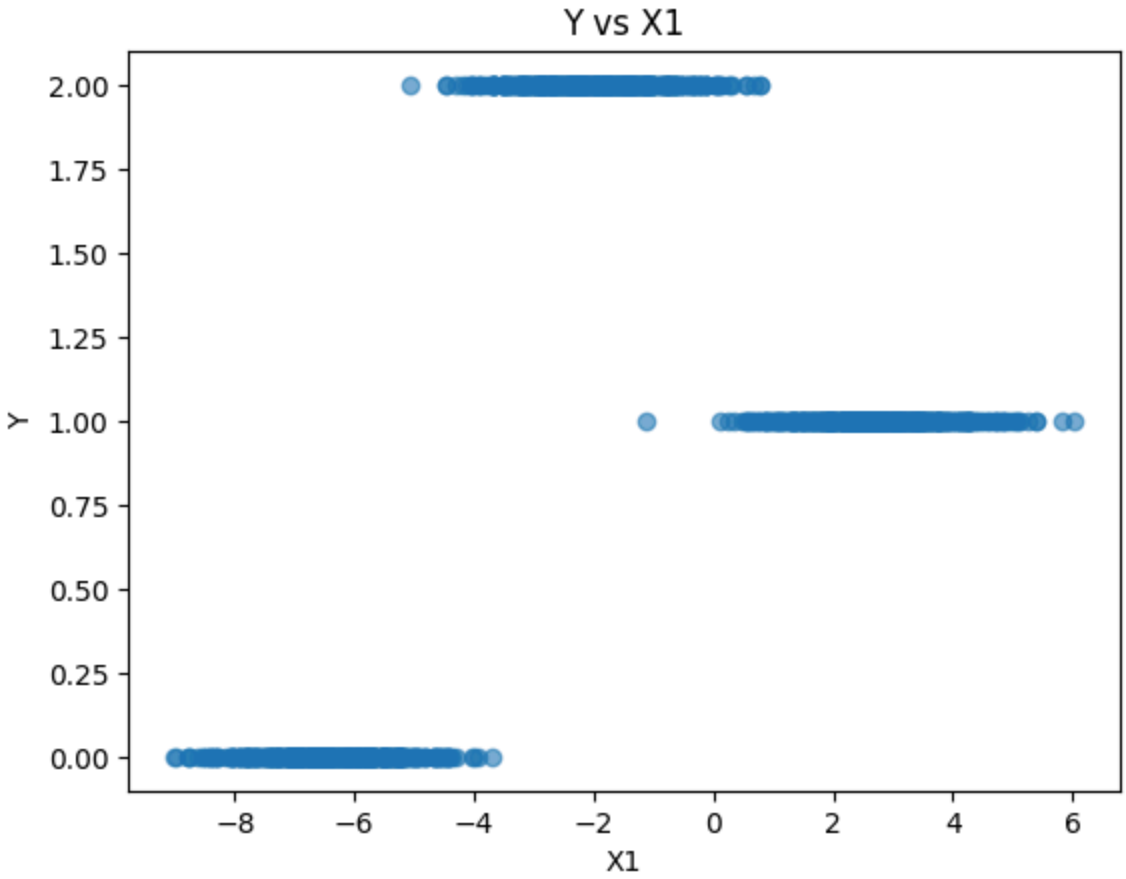
In [11]:
```
df_class.shape
```

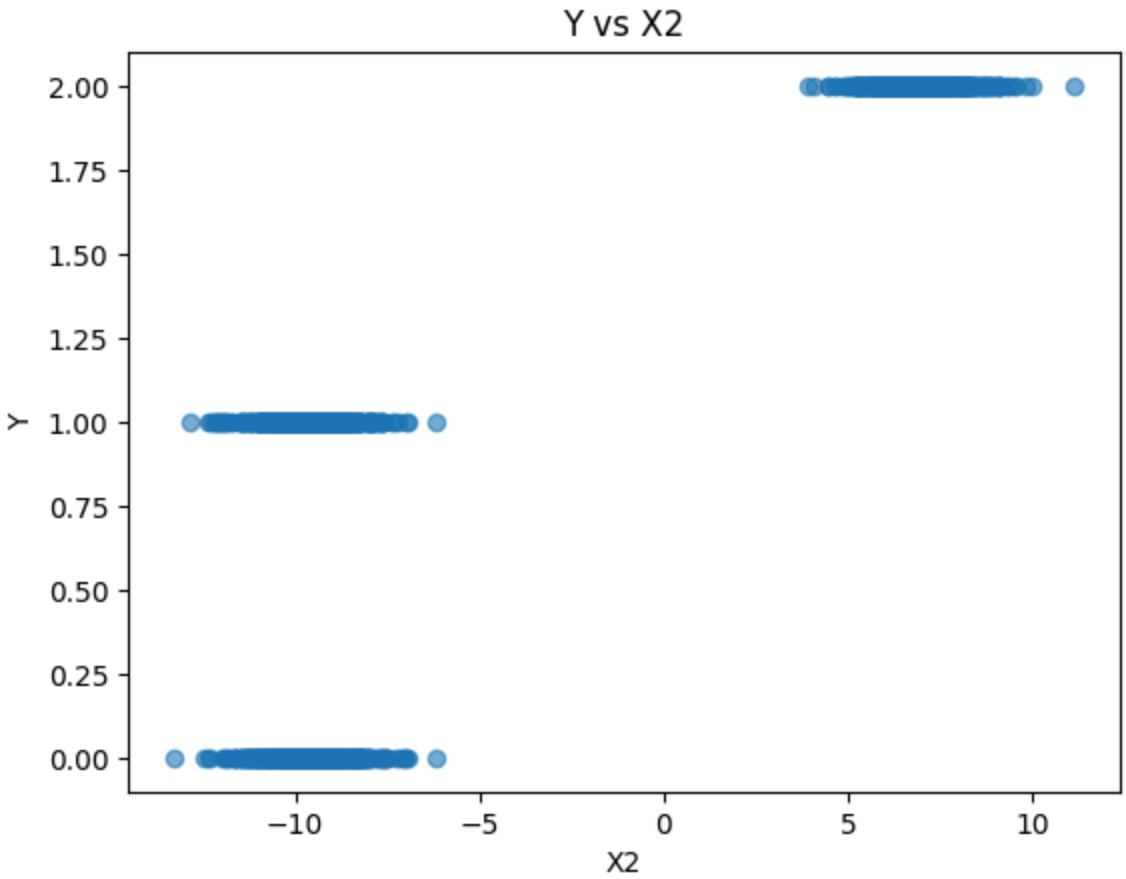
Out[11]:
```
(1485, 16)
```

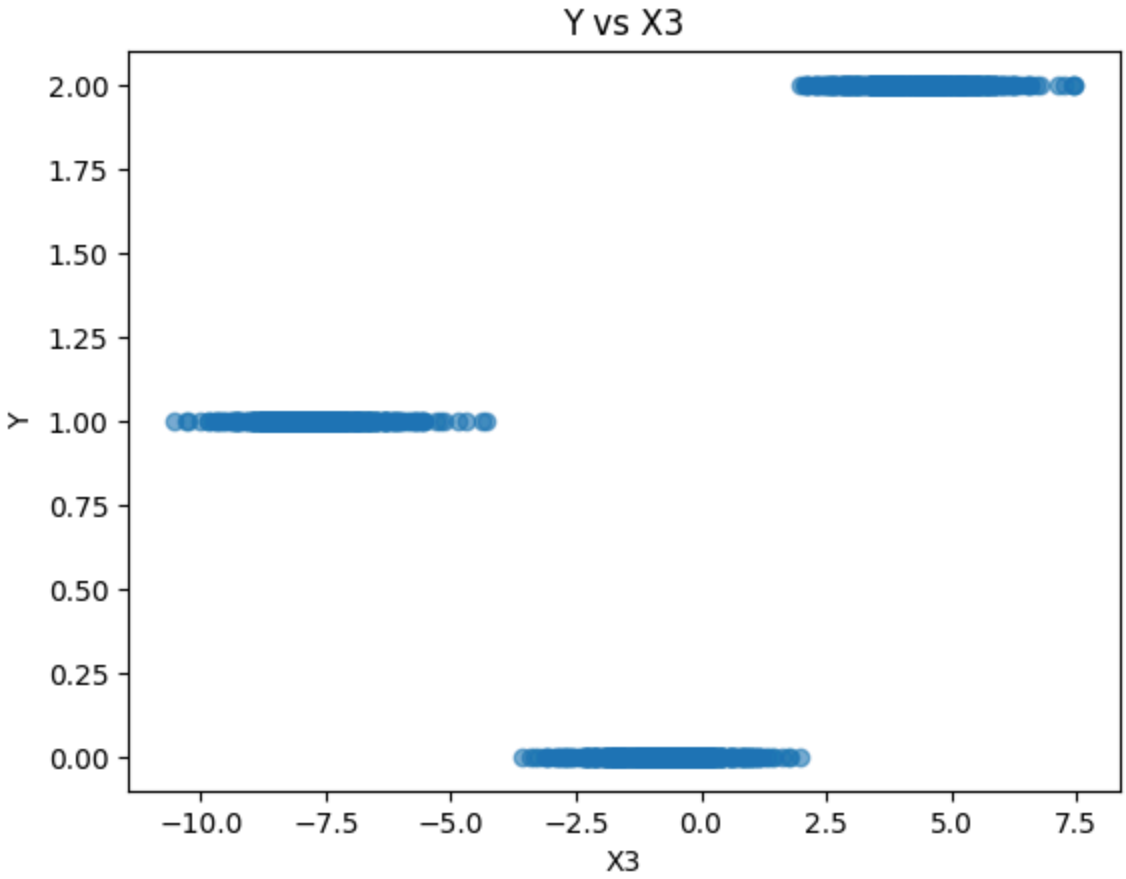
In [12]:
```
num_cols_class = df_class.select_dtypes(include=[np.number]).columns
df_class[num_cols_class].hist(bins=20, figsize=(10, 8))
plt.tight_layout()
plt.show()
```

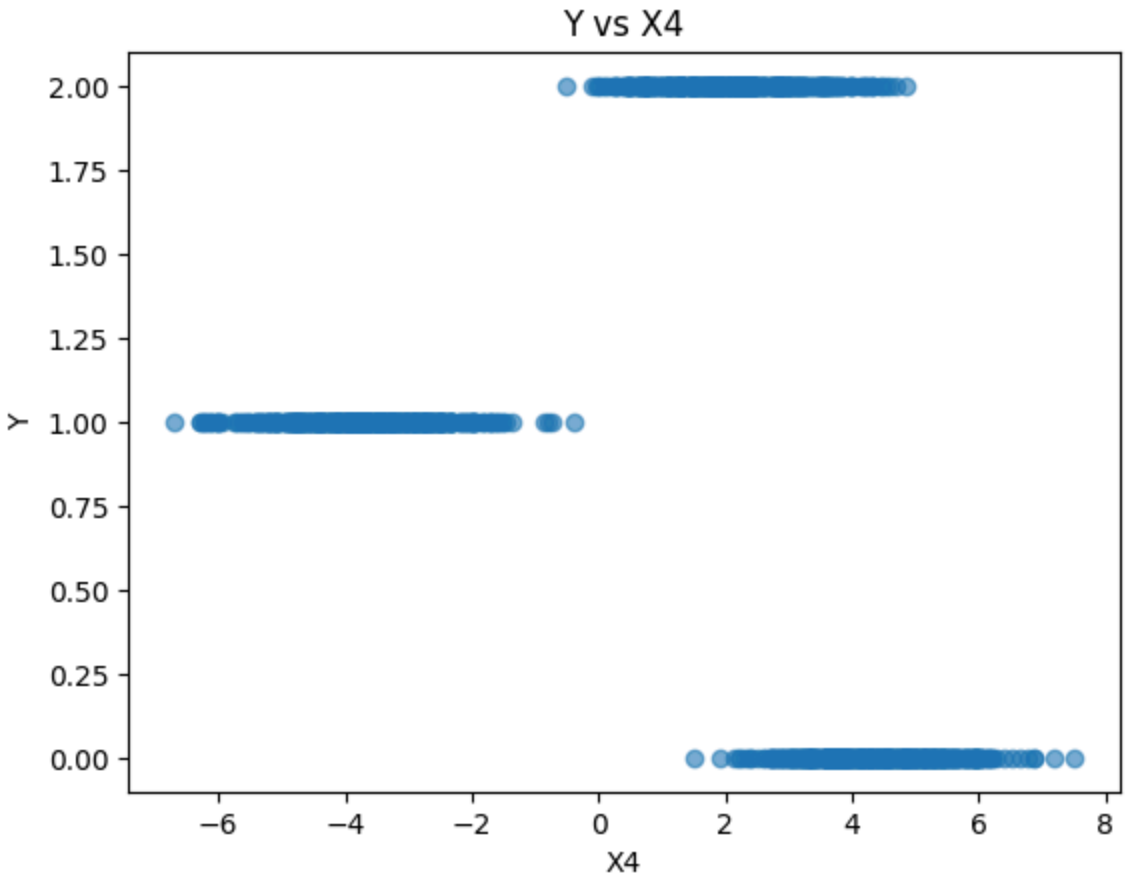


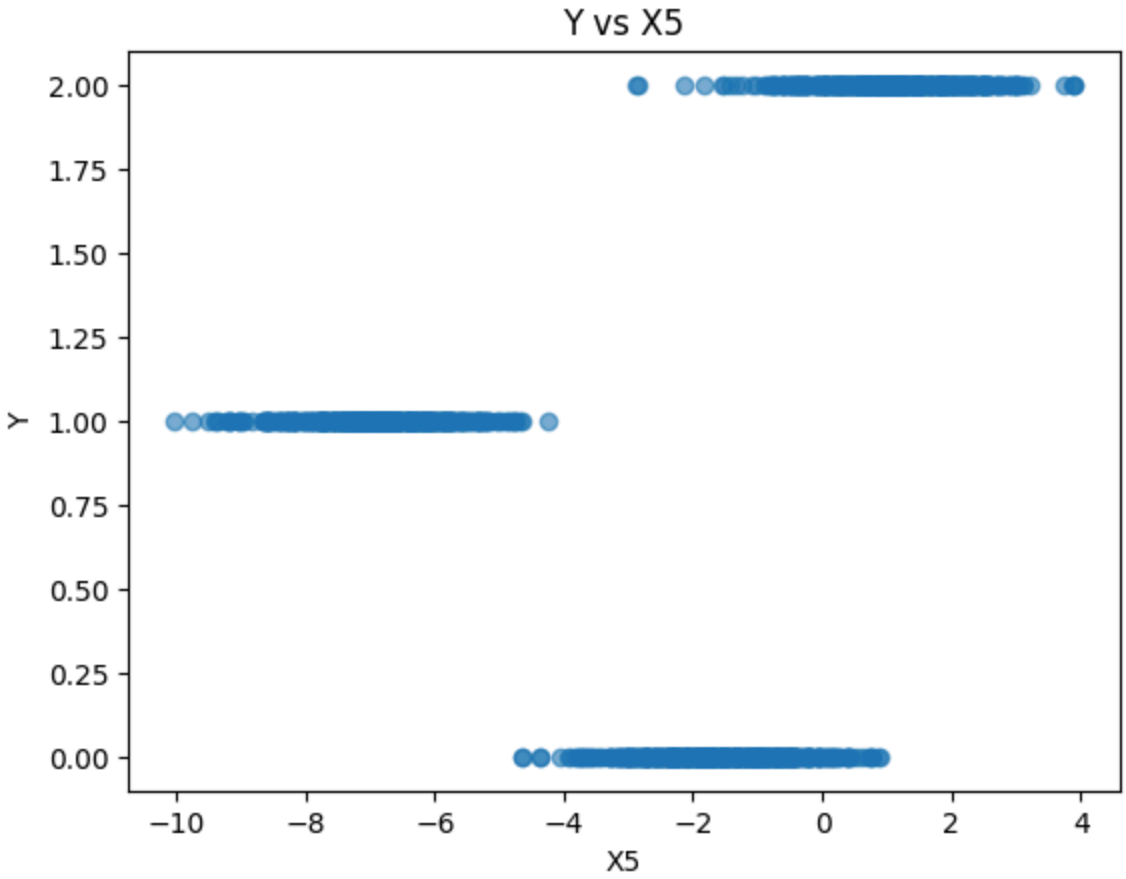
In [13]:
```
for col in [c for c in num_cols_class.to_list() if c != "Y"]:
    plt.figure()
    plt.scatter(df_class[col], df_class["Y"], alpha=0.6)
    plt.title(f"Y vs {col}")
    plt.xlabel(col)
    plt.ylabel("Y")
    plt.show()
```

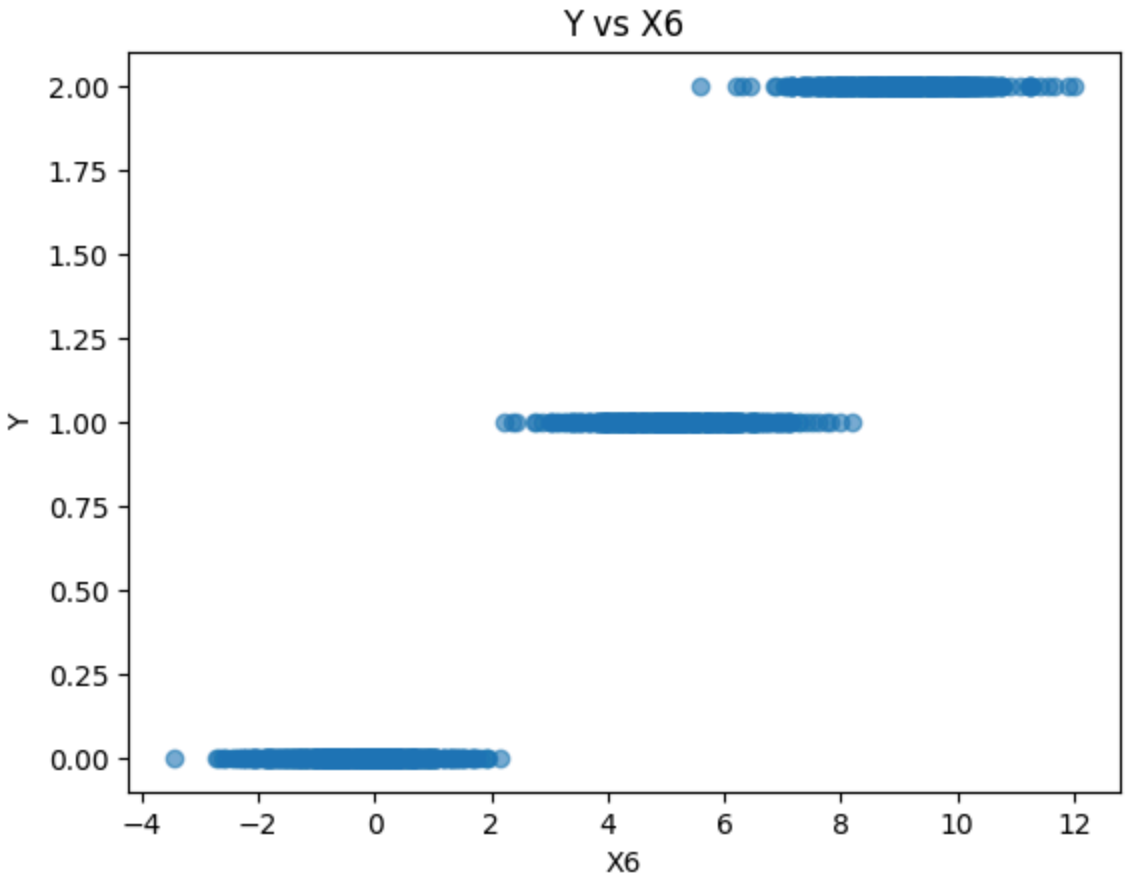

Y vs X1
Y
X1
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−8
−6
−4
−2
0
2
4
6

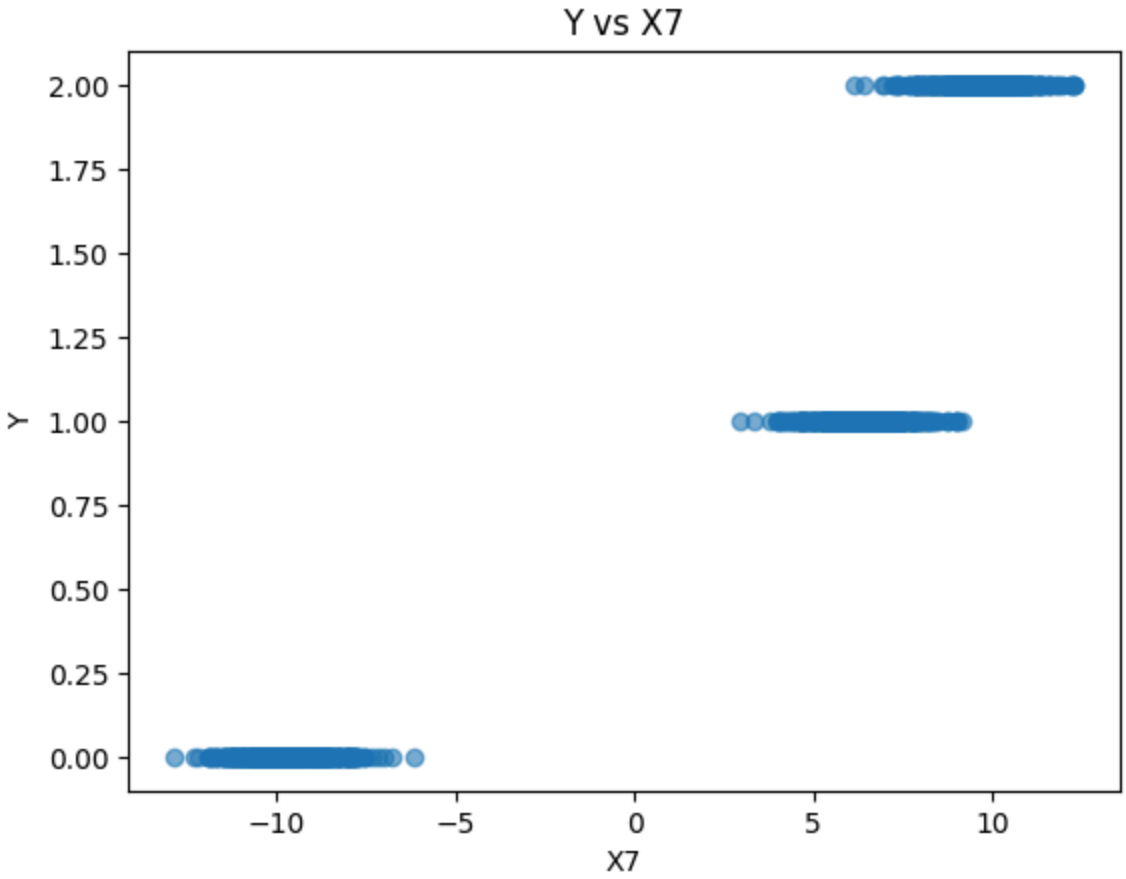

Y vs X2
Y
X2
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−10
−5
0
5
10

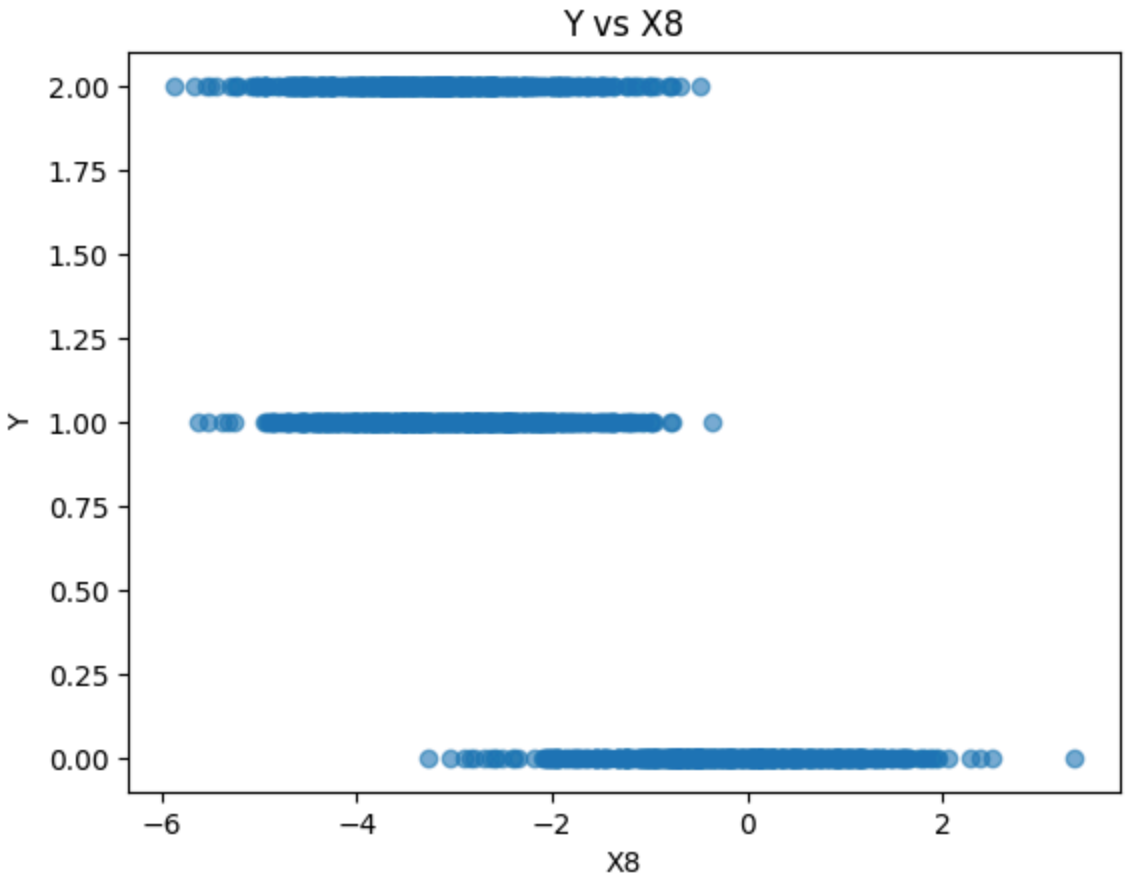

Y vs X3
Y
X3
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5

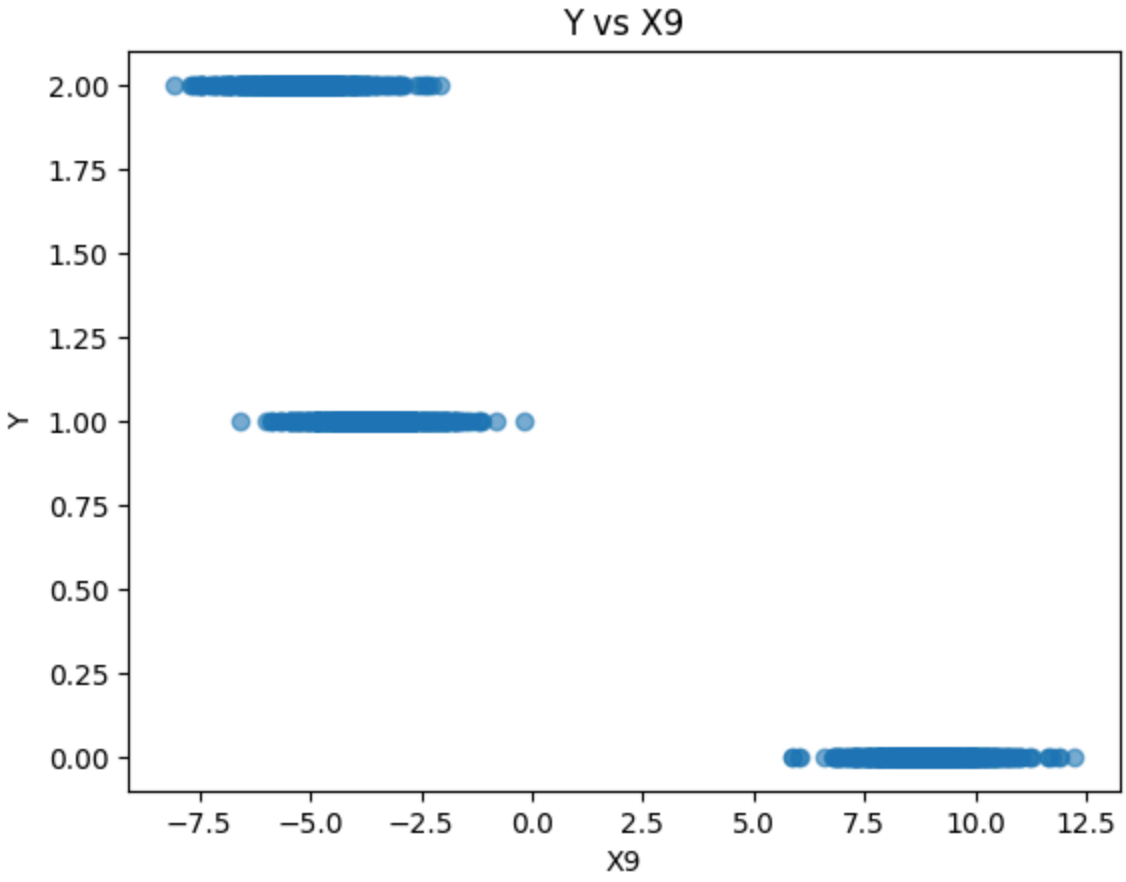

Y vs X4
Y
X4
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−6
−4
−2
0
2
4
6
8

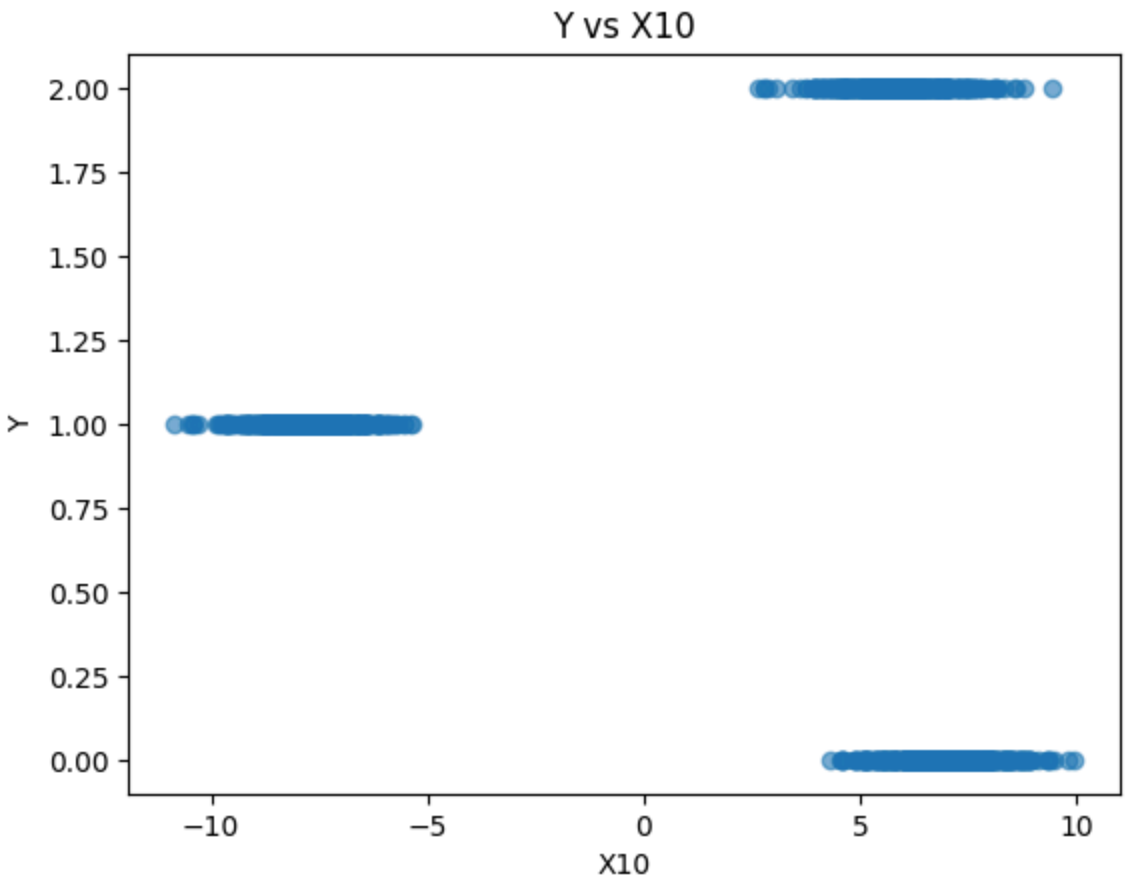

Y vs X5
Y
X5
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−10
−8
−6
−4
−2
0
2
4

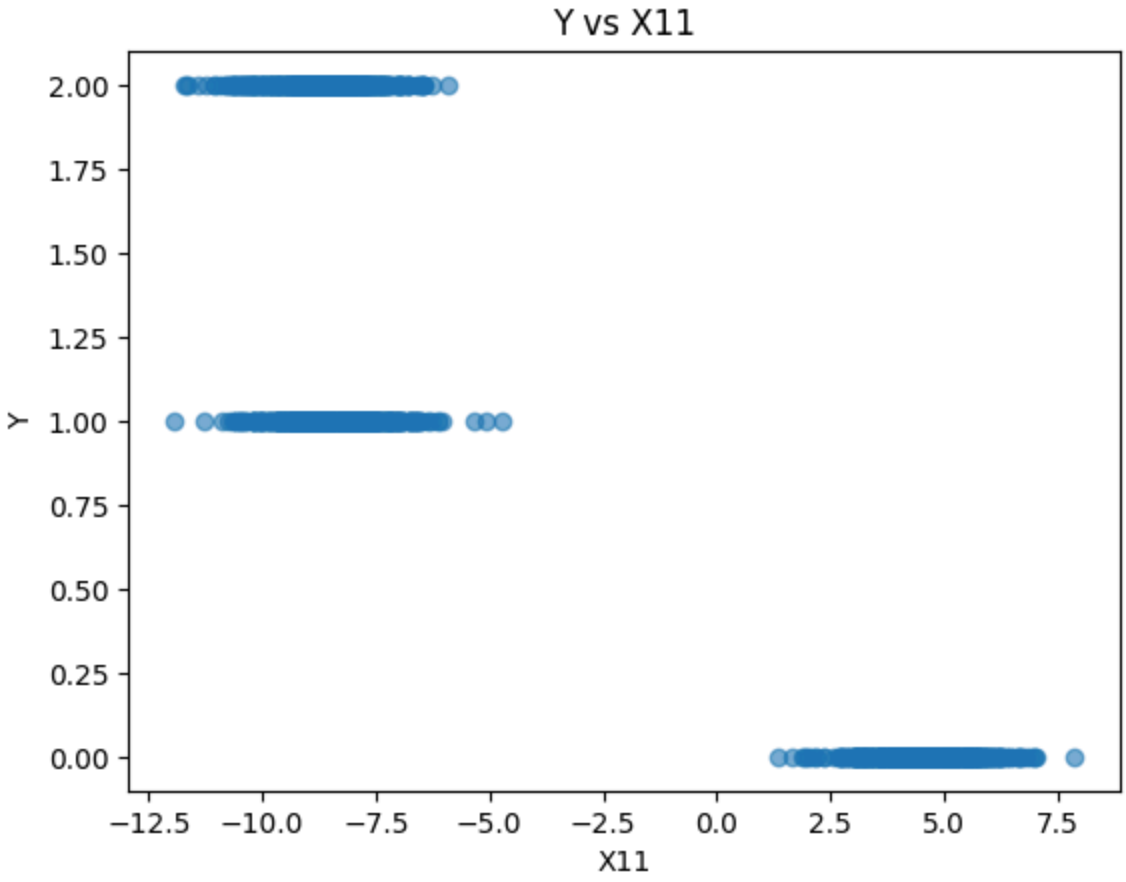

Y vs X6
Y
X6
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−4
−2
0
2
4
6
8
10
12

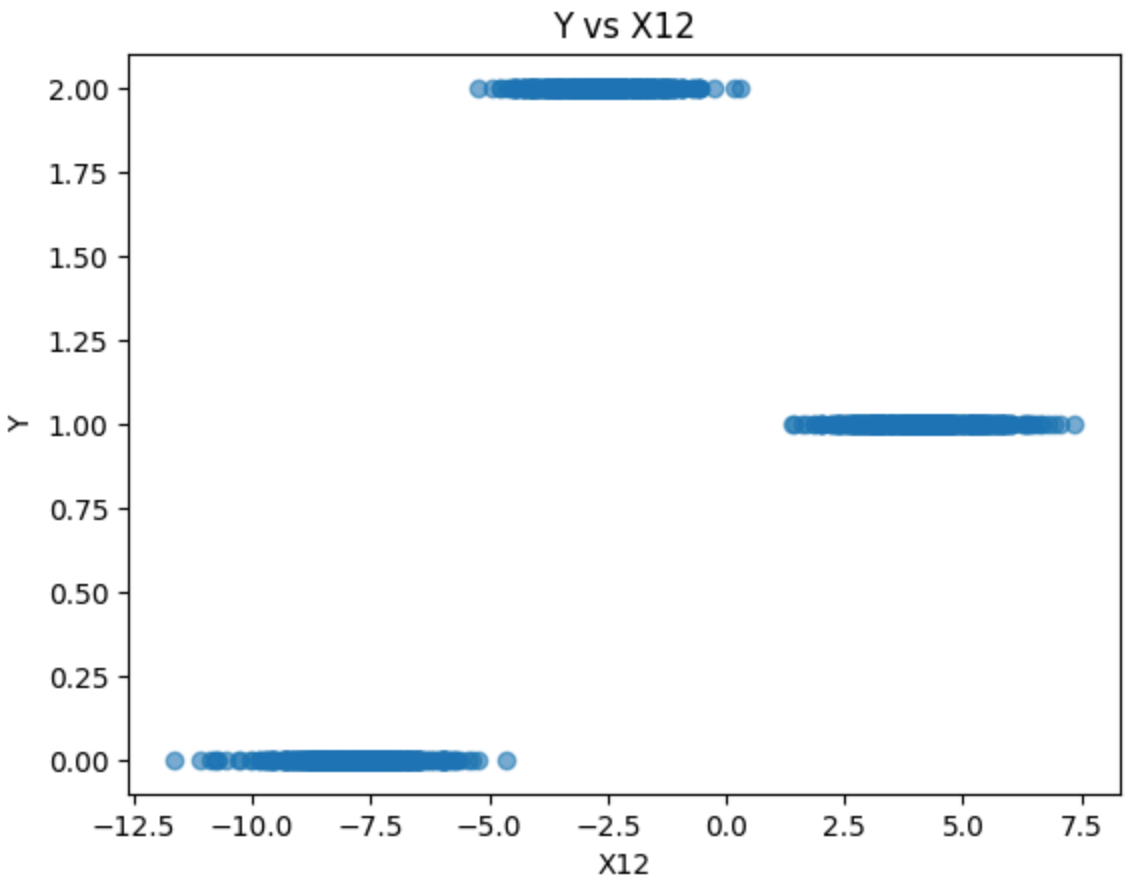

Y vs X7
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−10
−5
0
5
10
X7

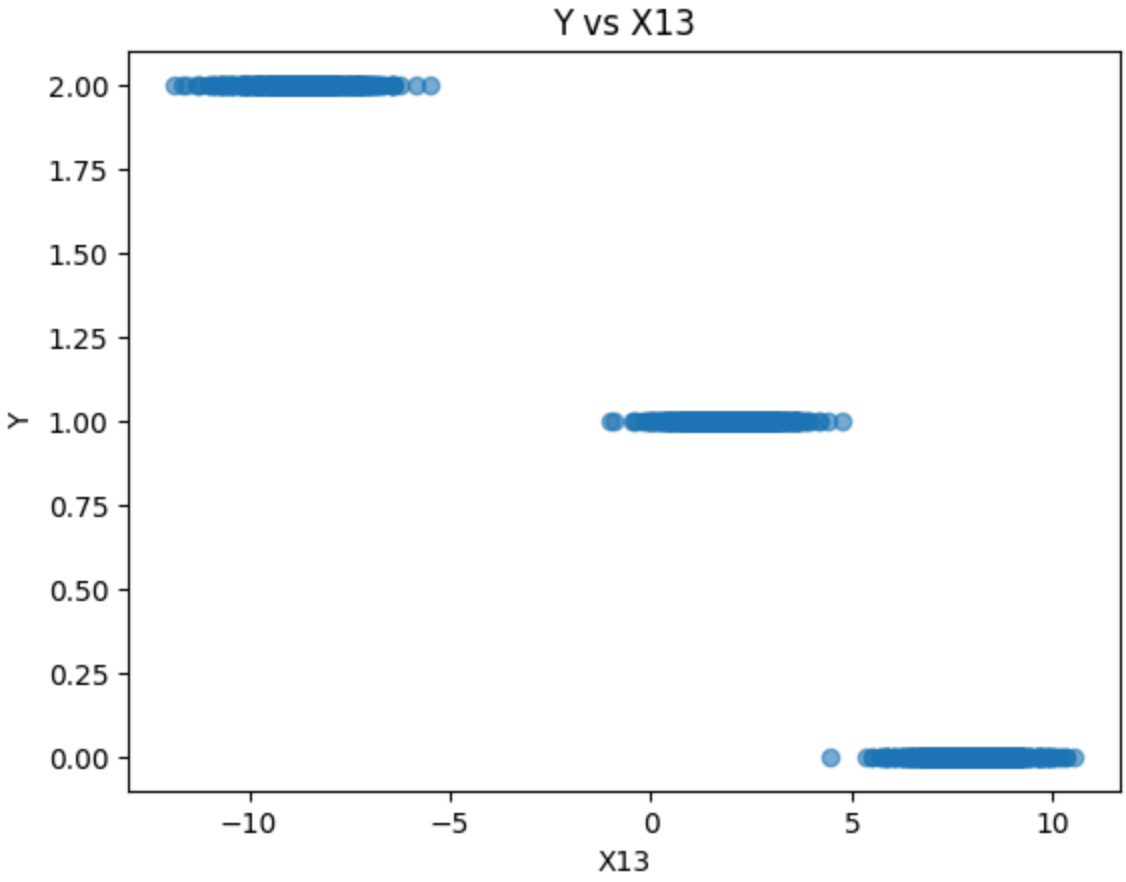

Y vs X8
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−6
−4
−2
0
2
X8

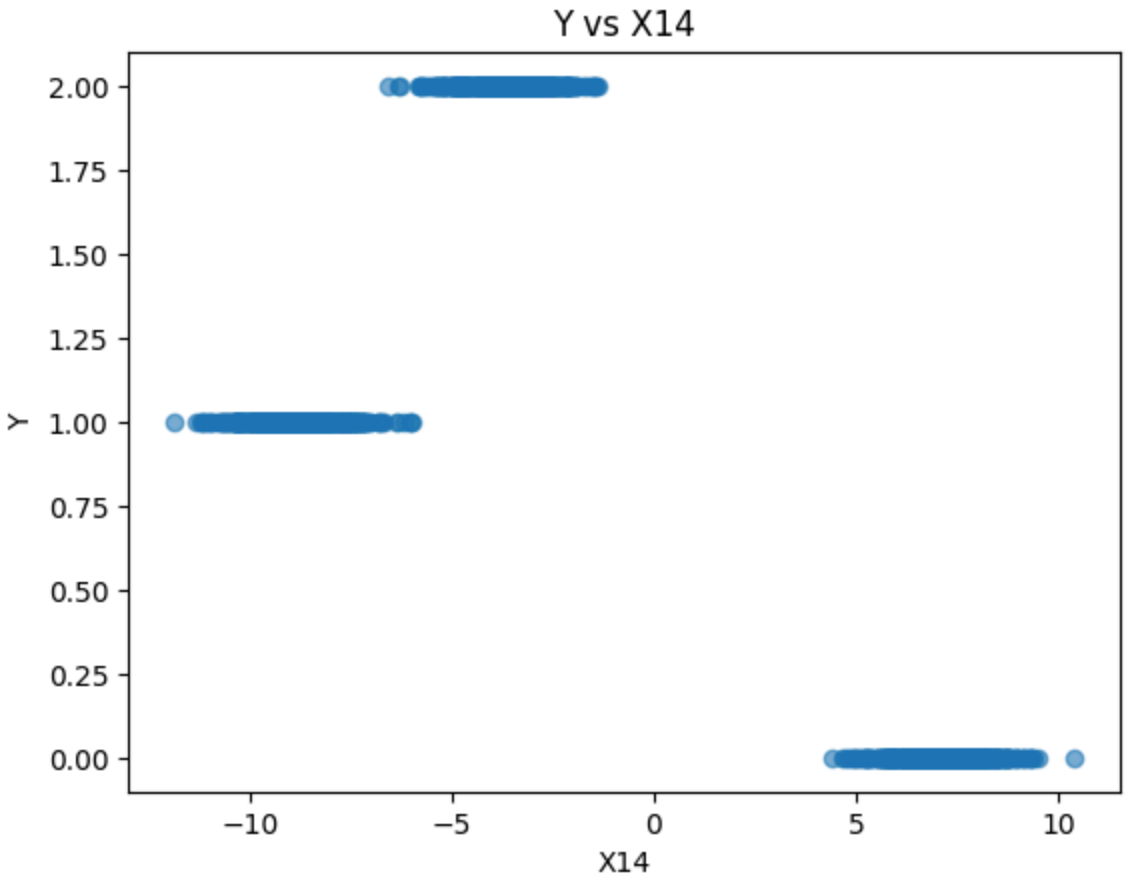

Y vs X9
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
X9

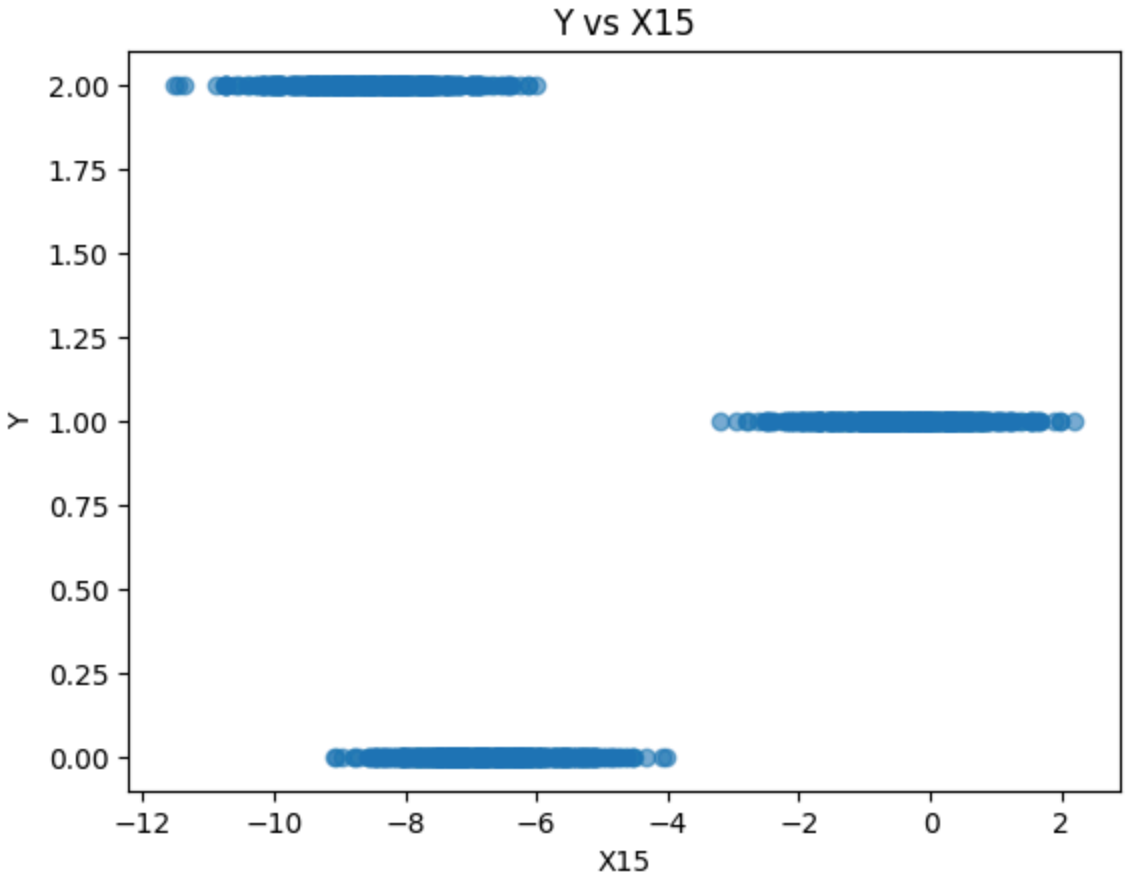

Y vs X10
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−10
−5
0
5
10
X10


Y vs X11
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
X11

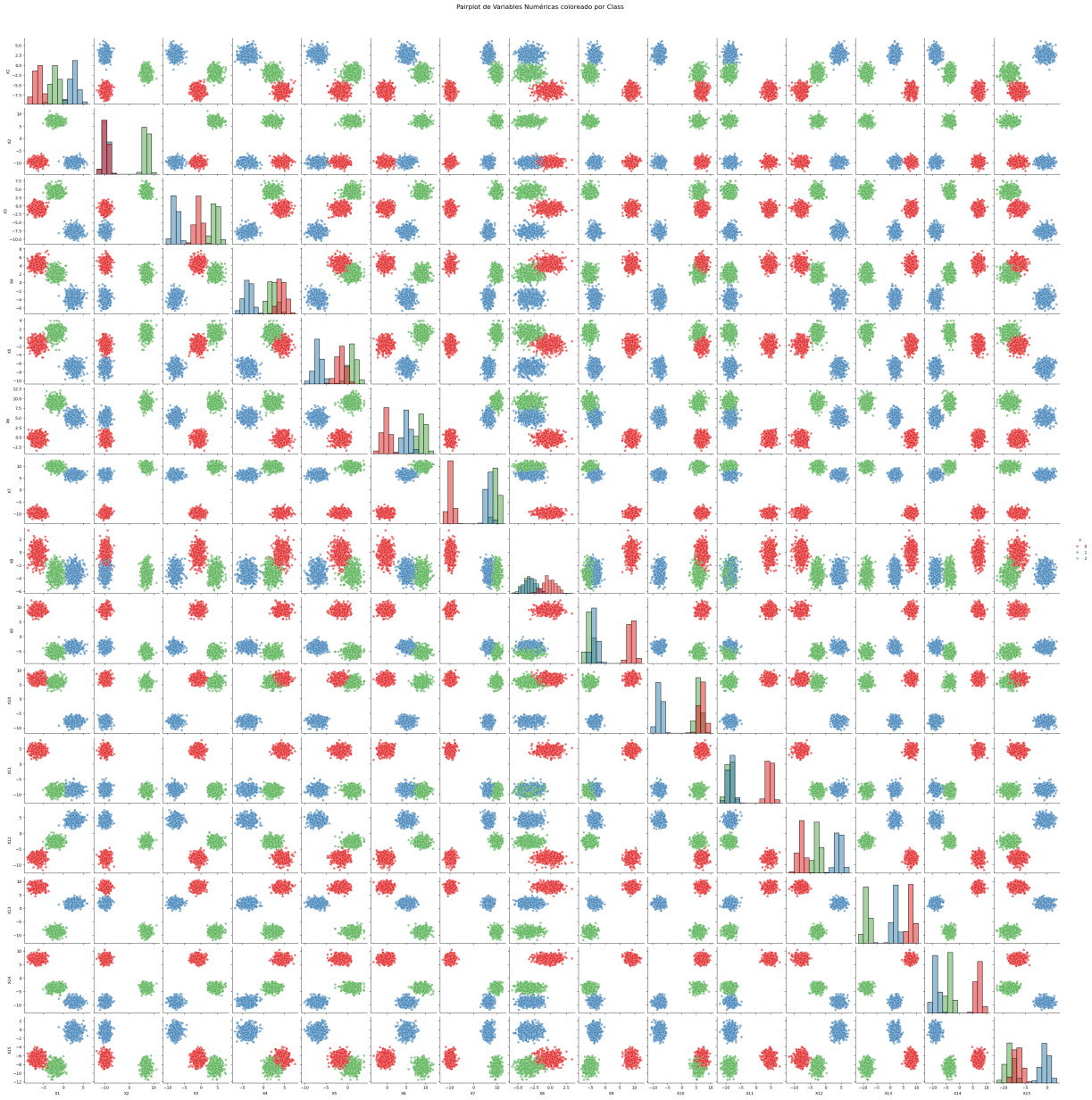

Y vs X12
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
X12


Y vs X13
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−10
−5
0
5
10
X13


Y vs X14
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Y
−10
−5
0
5
10
X14

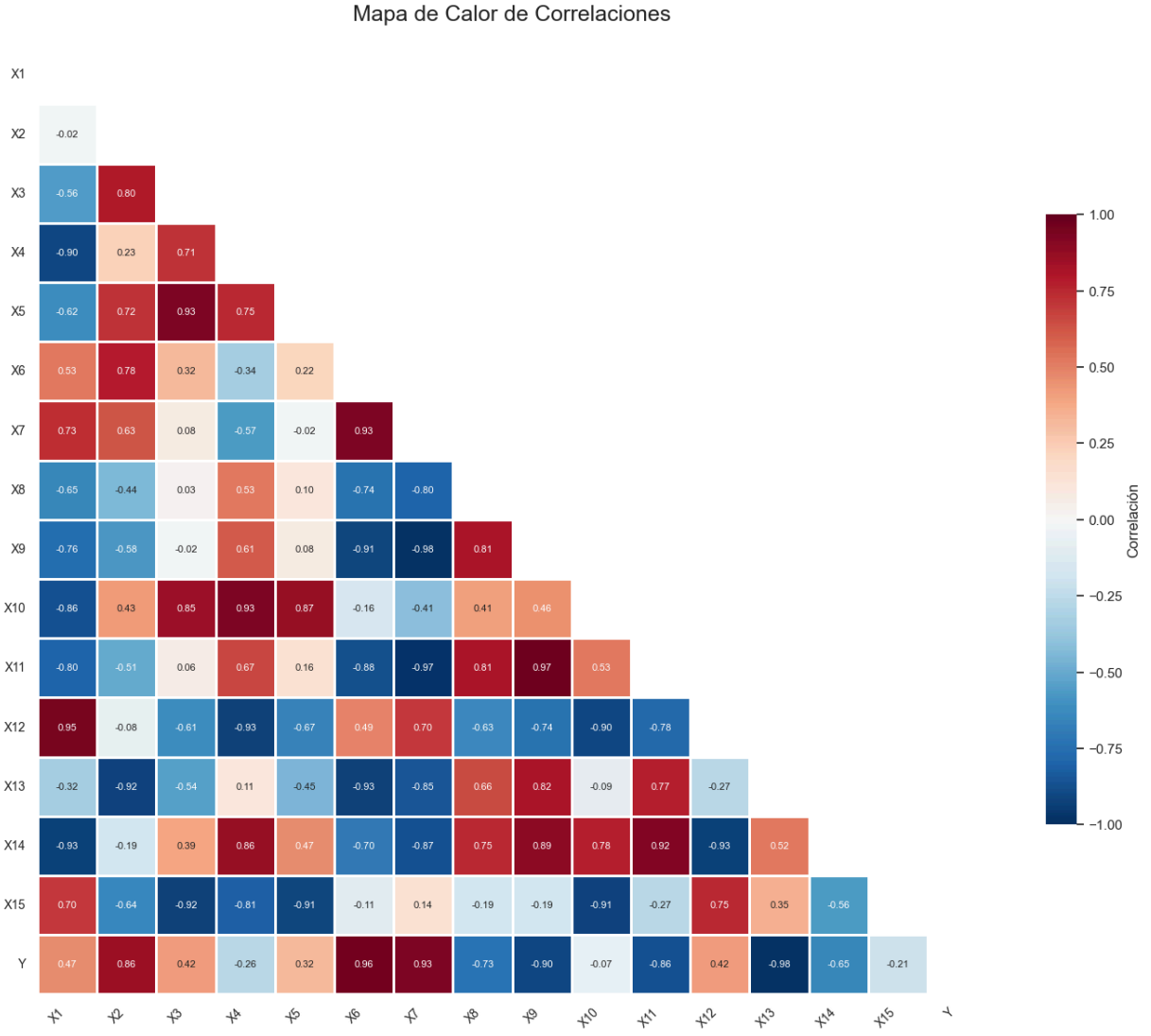


In [14]:
```
from pandas.plotting import scatter_matrix

numeric_cols_class = num_cols_class.drop("Y")
class_vals = df_class["Y"]

import seaborn as sns

sns.pairplot(
    df_class[numeric_cols_class.tolist() + ["Y"]],
    hue="Y",
    diag_kind="hist",
    palette="Set1",
    plot_kws={"alpha": 0.6, "marker": "o"},
)
plt.suptitle("Pairplot de Variables Numéricas coloreado por Class", y=1.02,
plt.show()
```



In [15]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

corr = df_class[numeric_cols_class.tolist() + ["Y"]].corr()


sns.set_theme(style="white")

mask = np.triu(np.ones_like(corr, dtype=bool))

n = corr.shape[0]
fig, ax = plt.subplots(figsize=(1.2 * n, 1.2 * n))

sns.heatmap(
    corr,
    mask=mask,
    annot=True,
    fmt=".2f",
```

```
    cmap="RdBu_r",
    center=0,
    linewidths=1.0,
    square=True,
    annot_kws={"size": 8},
    cbar_kws={"shrink": 0.6, "label": "Correlación"},
    vmin=-1,
    vmax=1,
    ax=ax,
)

ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha="right", fontsize=1
ax.set_yticklabels(ax.get_yticklabels(), rotation=0, fontsize=10)

plt.subplots_adjust(bottom=0.3, left=0.2, top=0.9)

ax.set_title("Mapa de Calor de Correlaciones", fontsize=18, pad=20)

plt.show()
```



Regression

In [16]:
```
df_reg = pd.read_csv("../data/regression.csv")
```

In [17]:
```
df_reg.shape
```

Out[17]:
```
(1500, 13)
```

In [18]:
```
df_reg.dtypes
```

Out[18]:
```
X1      float64
X2      float64
X3      float64
X4      float64
X5      float64
X6      float64
X7      float64
X8      float64
X9      float64
X10     float64
X11     float64
X12     float64
Y       float64
dtype: object
```

In [19]:
```
df_reg.head()
```

Out[19]:

	X1	X2	X3	X4	X5	X6	
0	5468.737967	1.703822	-8613.627601	0.284719	-1.585462	-0.554724	-1.0230
1	-13104.449210	-1.019682	23061.174510	-0.820498	-0.265240	2.099421	-0.3163
2	6204.272157	0.734018	133965.765100	-0.710429	0.574379	0.511245	0.2981
3	23979.672370	-1.777481	45514.777250	-0.148987	-0.006399	0.464724	0.3568
4	-2390.952073	-2.418137	47102.194950	0.384587	0.359033	0.178320	-1.0630

In [20]:
```
df_reg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X1      1499 non-null   float64
 1   X2      1500 non-null   float64
 2   X3      1495 non-null   float64
 3   X4      1497 non-null   float64
 4   X5      1495 non-null   float64
 5   X6      1498 non-null   float64
 6   X7      1499 non-null   float64
 7   X8      1498 non-null   float64
 8   X9      1500 non-null   float64
 9   X10     1496 non-null   float64
 10  X11     1498 non-null   float64
 11  X12     1499 non-null   float64
 12  Y       1499 non-null   float64
dtypes: float64(13)
memory usage: 152.5 KB
```

In [21]:
```
df_reg.describe()
```

Out[21]:

	X1	X2	X3	X4	X5	
count	1499.000000	1500.000000	1495.000000	1497.000000	1495.000000	1498
mean	-298.970046	-0.000182	2505.655649	0.019644	-0.037554	-0
std	12664.656465	1.036299	95364.185732	1.017798	0.975190	1
min	-39838.521840	-3.710679	-336700.253400	-2.796599	-3.282985	-3
25%	-9433.010398	-0.677534	-60485.305915	-0.694018	-0.698821	-0
50%	-496.180129	0.021941	1748.071119	0.027749	-0.034482	-0
75%	8318.518644	0.705290	70803.472755	0.728685	0.615946	0
max	44114.014510	3.156200	312030.703900	3.344649	3.515152	3

In [22]:
```
missing_summary = df_reg.isnull().sum().to_frame(name="n_missing")
missing_summary["pct_missing"] = missing_summary["n_missing"] / len(df_reg)
print("\nResumen de valores faltantes:")
missing_summary
```

```
Resumen de valores faltantes:
```

Out[22]:

	n_missing	pct_missing
X1	1	0.066667
X2	0	0.000000
X3	5	0.333333
X4	3	0.200000
X5	5	0.333333
X6	2	0.133333
X7	1	0.066667
X8	2	0.133333
X9	0	0.000000
X10	4	0.266667
X11	2	0.133333
X12	1	0.066667
Y	1	0.066667

In [23]:
```
df_reg.shape
```

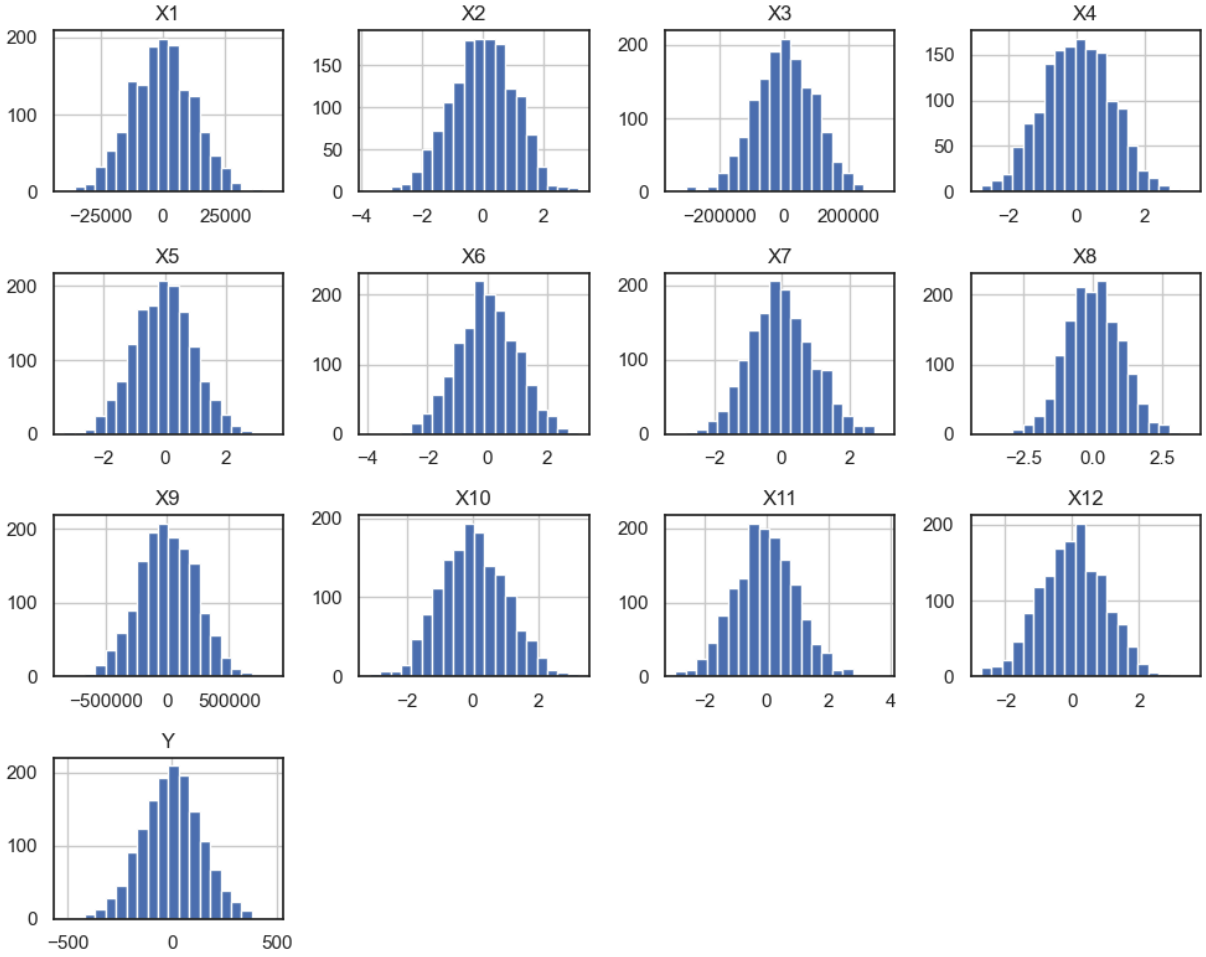
Out[23]:
```
(1500, 13)
```

In [24]:
```
df_reg = df_reg.dropna()
```

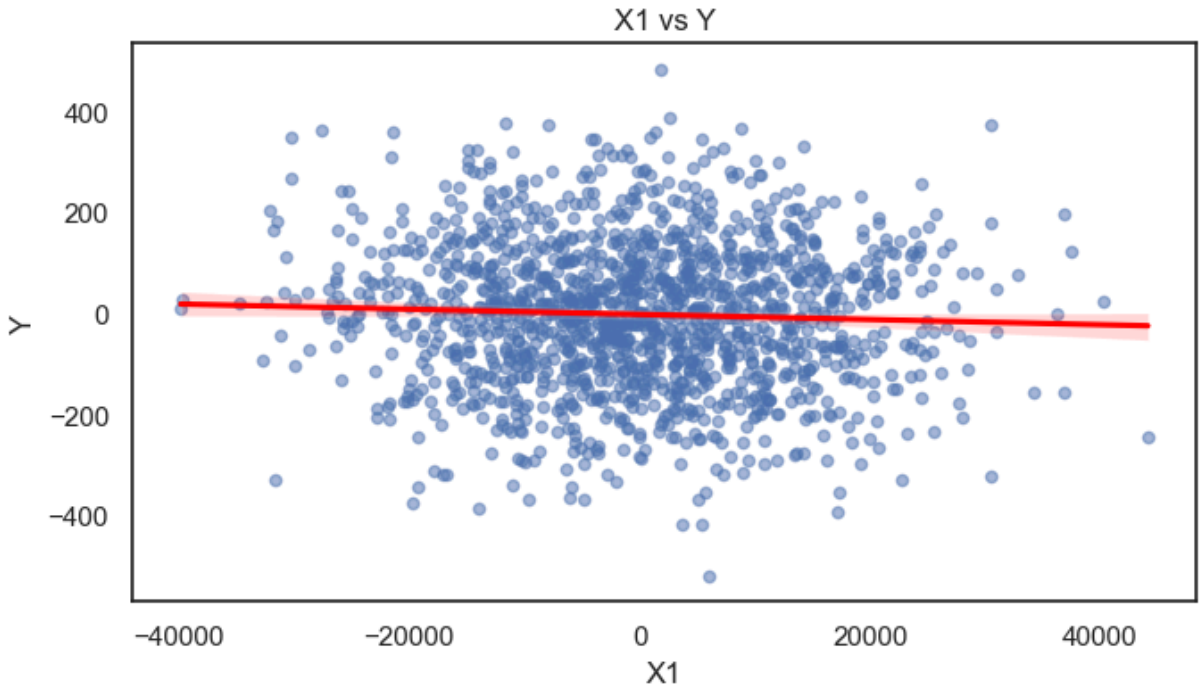
In [25]:
```
df_reg.shape
```

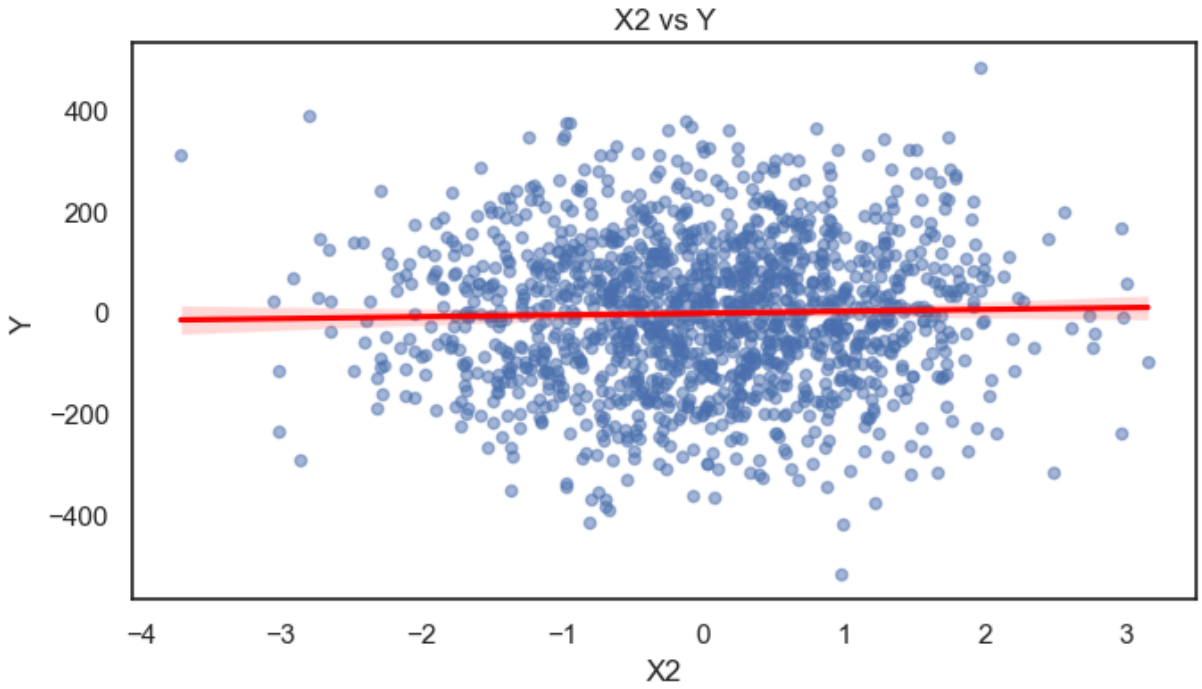
Out[25]:
```
(1473, 13)
```

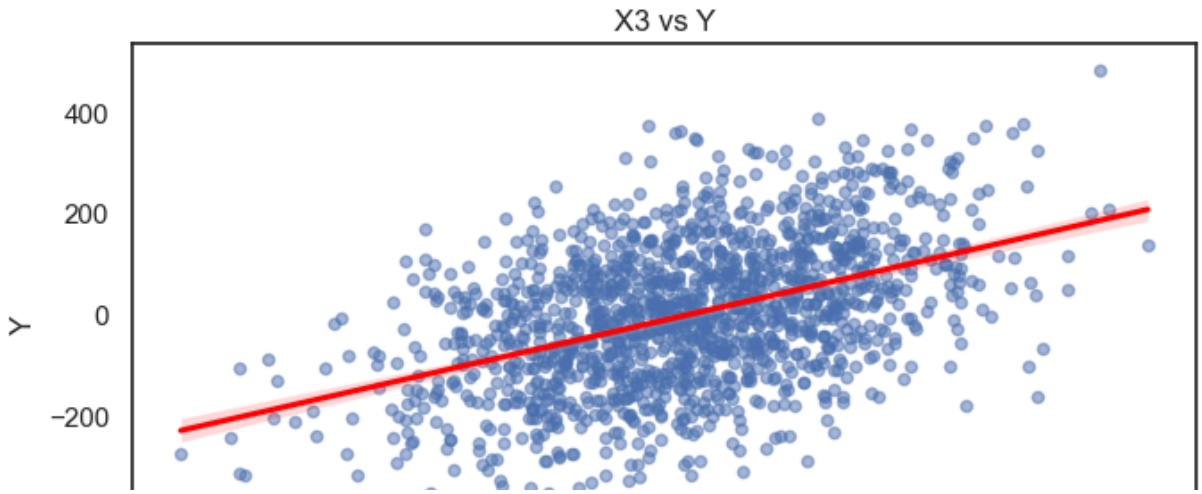
In [26]:
```
num_cols_reg = df_reg.select_dtypes(include=[np.number]).columns
df_reg[num_cols_reg].hist(bins=20, figsize=(10, 8))
plt.tight_layout()
plt.show()
```

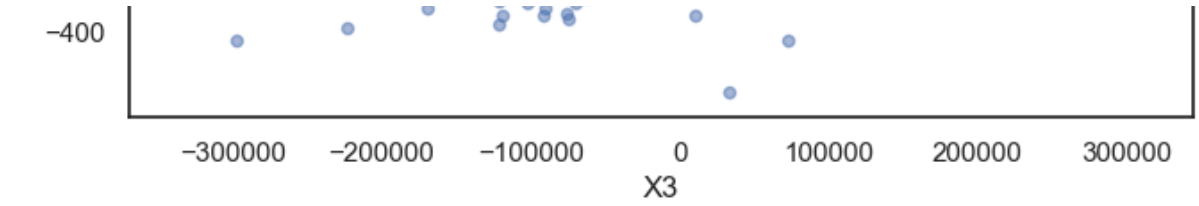


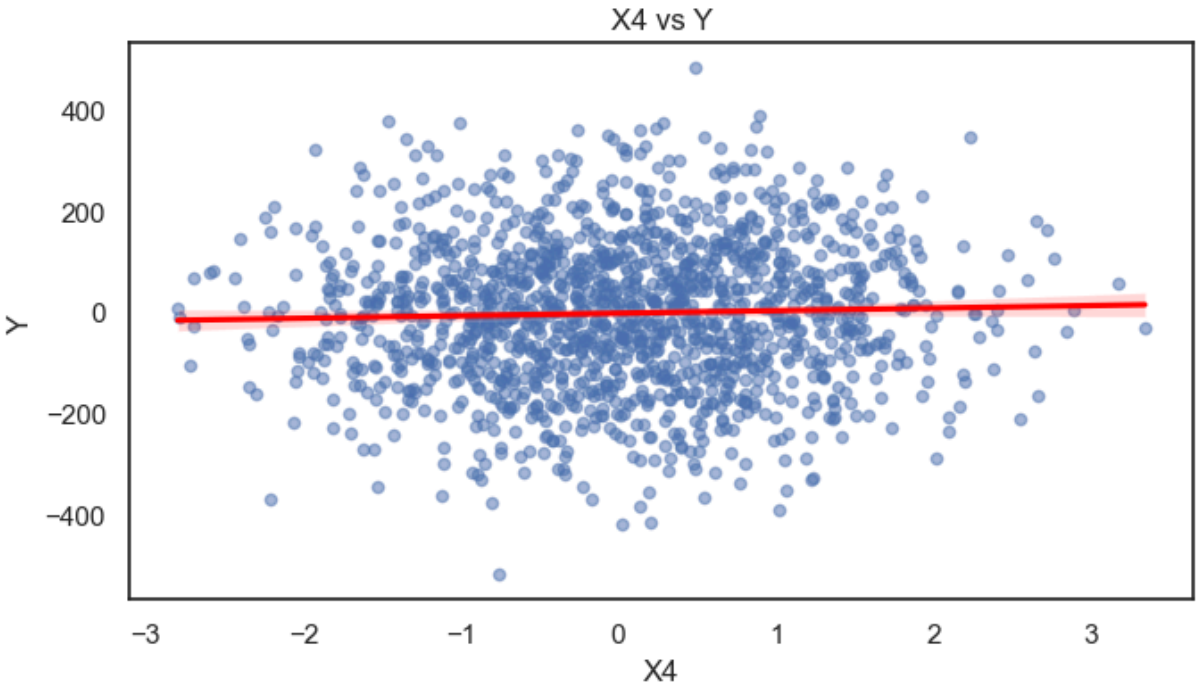
In [27]:
```
from pandas.plotting import scatter_matrix

numeric_cols_reg = num_cols_reg.drop("Y")
reg_vals = df_reg["Y"]

import matplotlib.pyplot as plt
import seaborn as sns

n_feats = len(numeric_cols_reg)
fig, axes = plt.subplots(n_feats, 1, figsize=(7, 4 * n_feats))

for ax, feat in zip(axes, numeric_cols_reg):
    sns.regplot(
        x=feat,
        y="Y",
        data=df_reg,
        ax=ax,
        scatter_kws={"alpha": 0.5, "s": 20},
        line_kws={"color": "red"},
    )
    ax.set_title(f"{feat} vs Y")
    ax.set_xlabel(feat)
    ax.set_ylabel("Y")

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

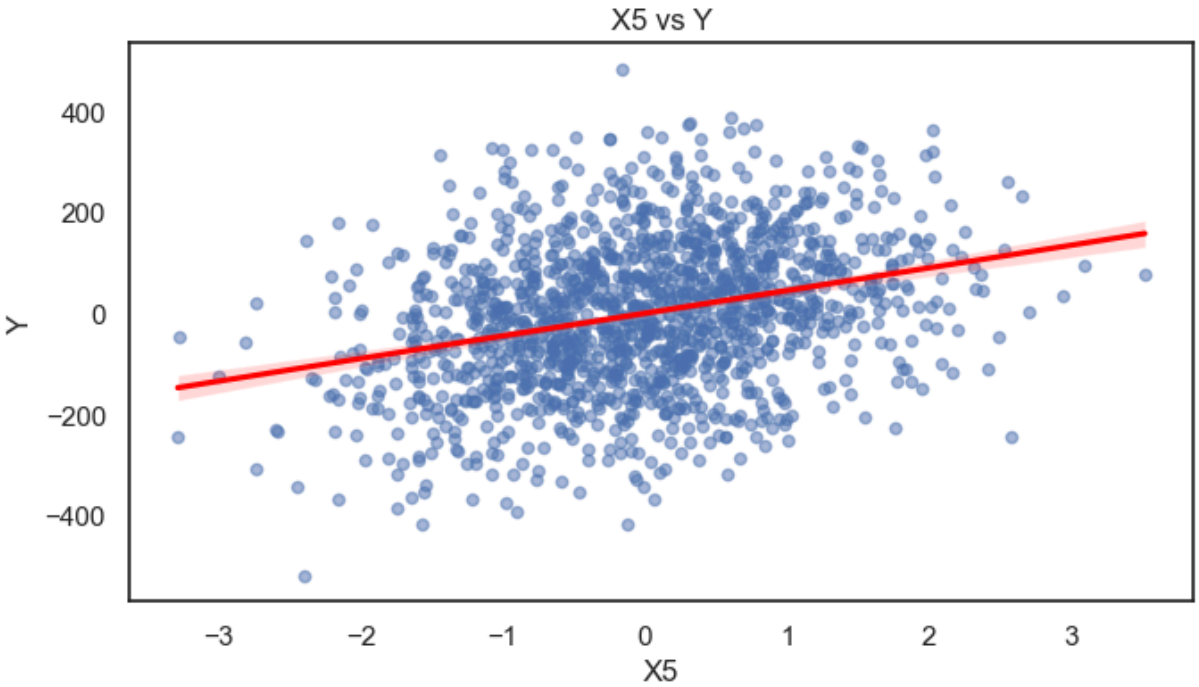

X1 vs Y
400
200
0
−200
−400
Y
−40000
−20000
0
20000
40000
X1

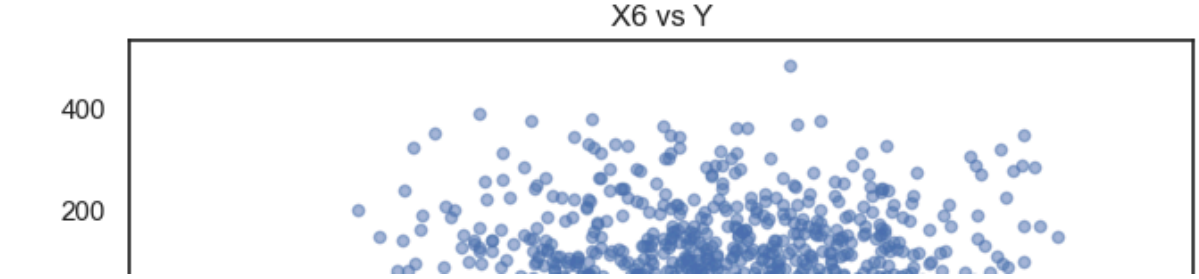

X2 vs Y
400
200
0
−200
−400
Y
−4
−3
−2
−1
0
1
2
3
X2

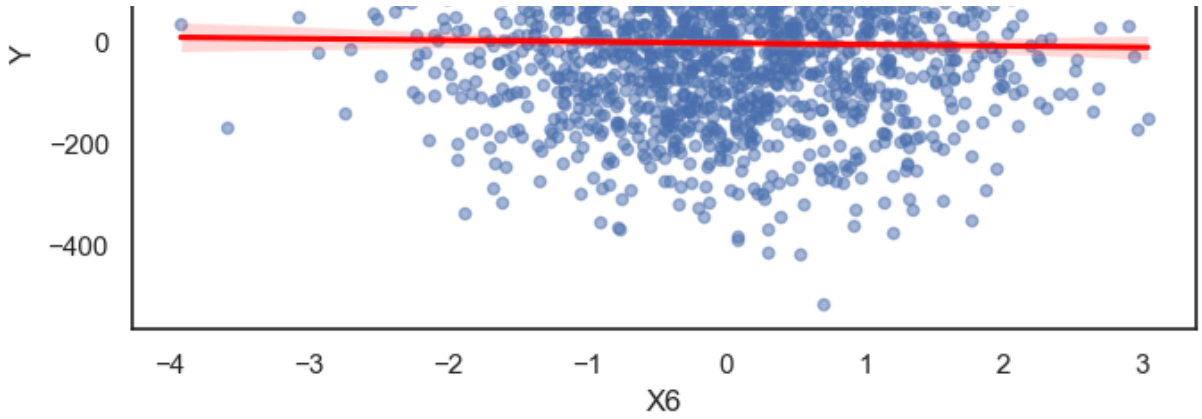

X3 vs Y
400
200
0
−200
Y

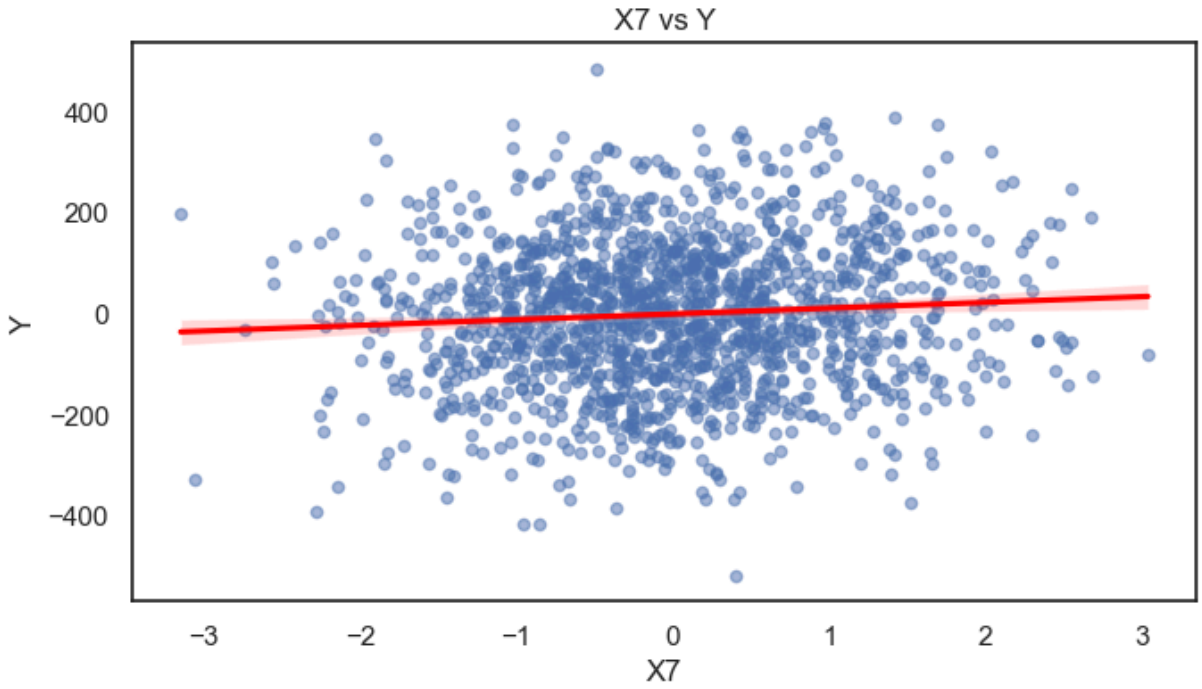

−400
−300000
−200000
−100000
0
100000
200000
300000
X3

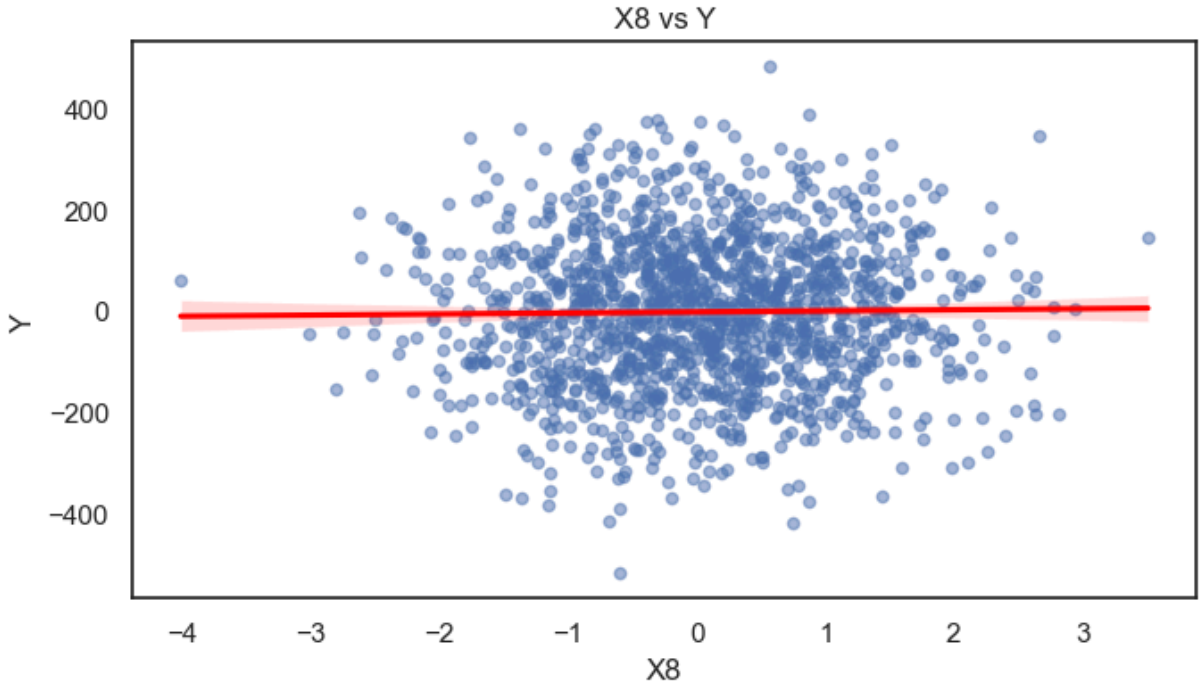

X4 vs Y
400
200
0
−200
−400
Y
−3
−2
−1
0
1
2
3
X4

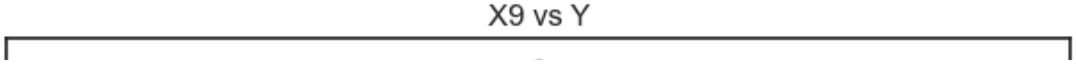

X5 vs Y
400
200
0
−200
−400
Y
−3
−2
−1
0
1
2
3
X5

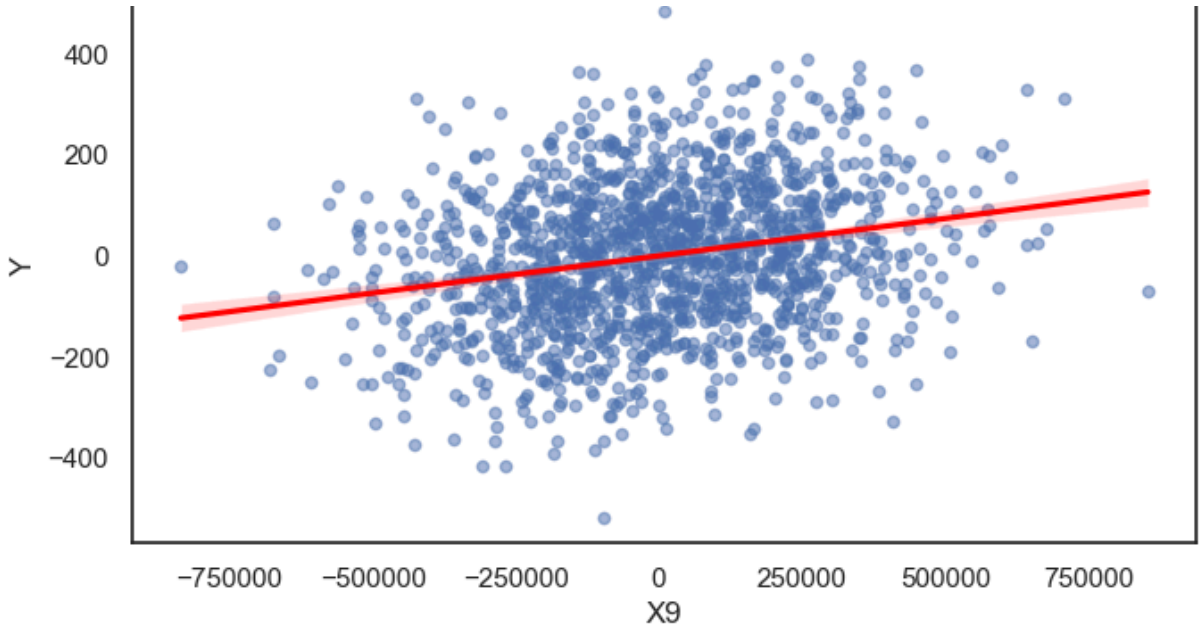

X6 vs Y
400
200

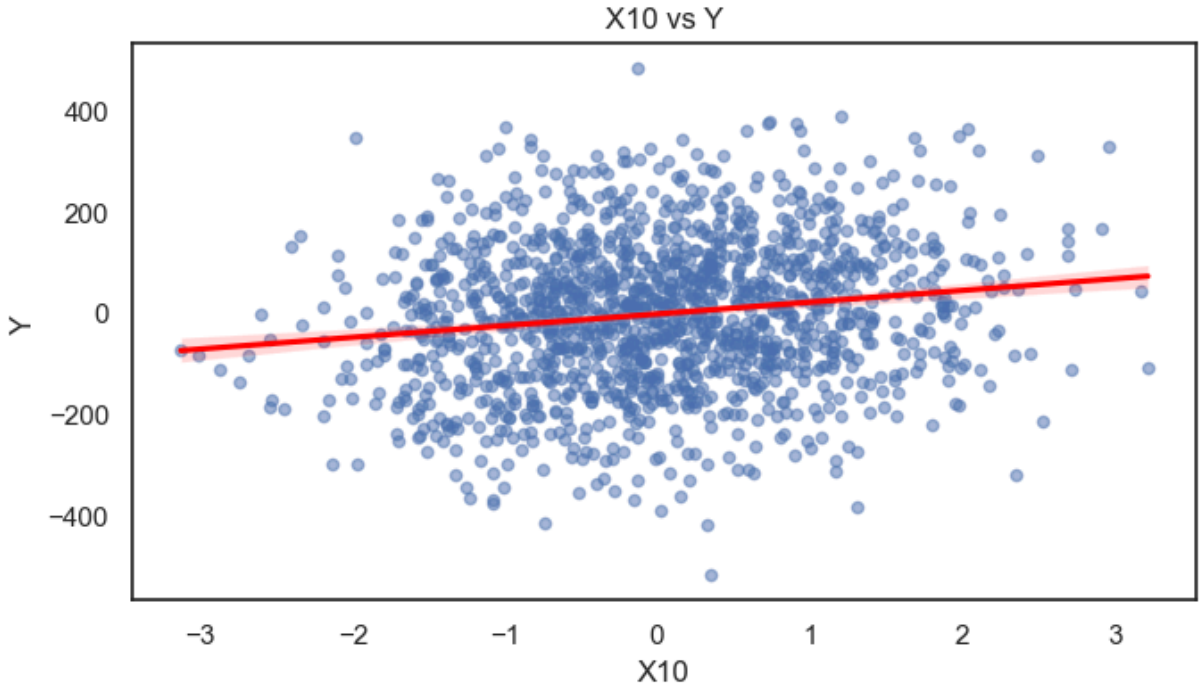

Y
0
−200
−400
−4
−3
−2
−1
0
1
2
3
X6

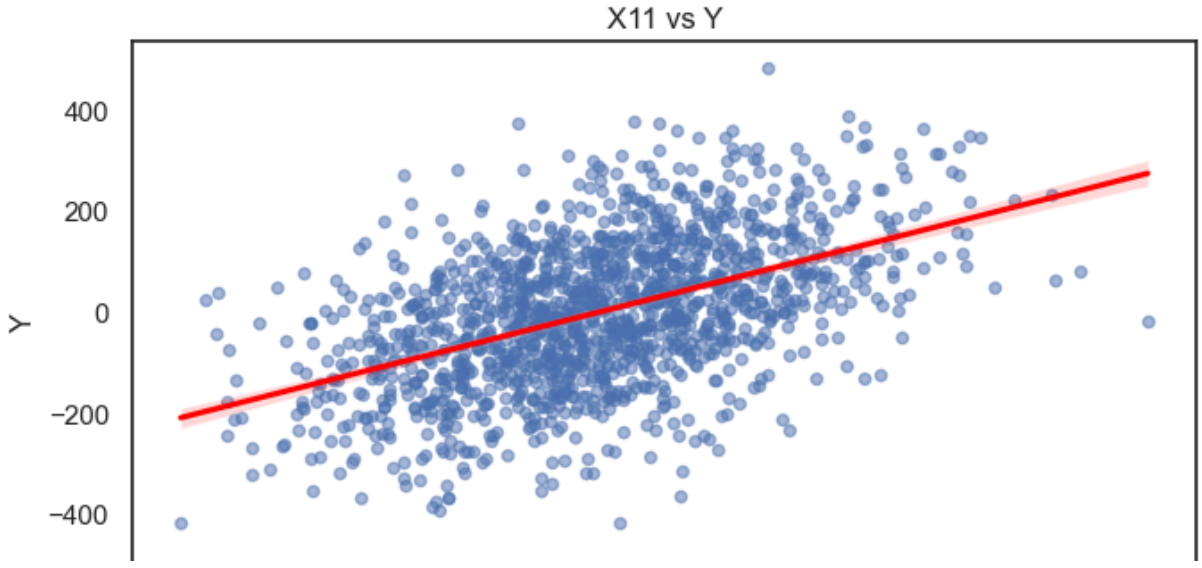

X7 vs Y
Y
400
200
0
−200
−400
−3
−2
−1
0
1
2
3
X7

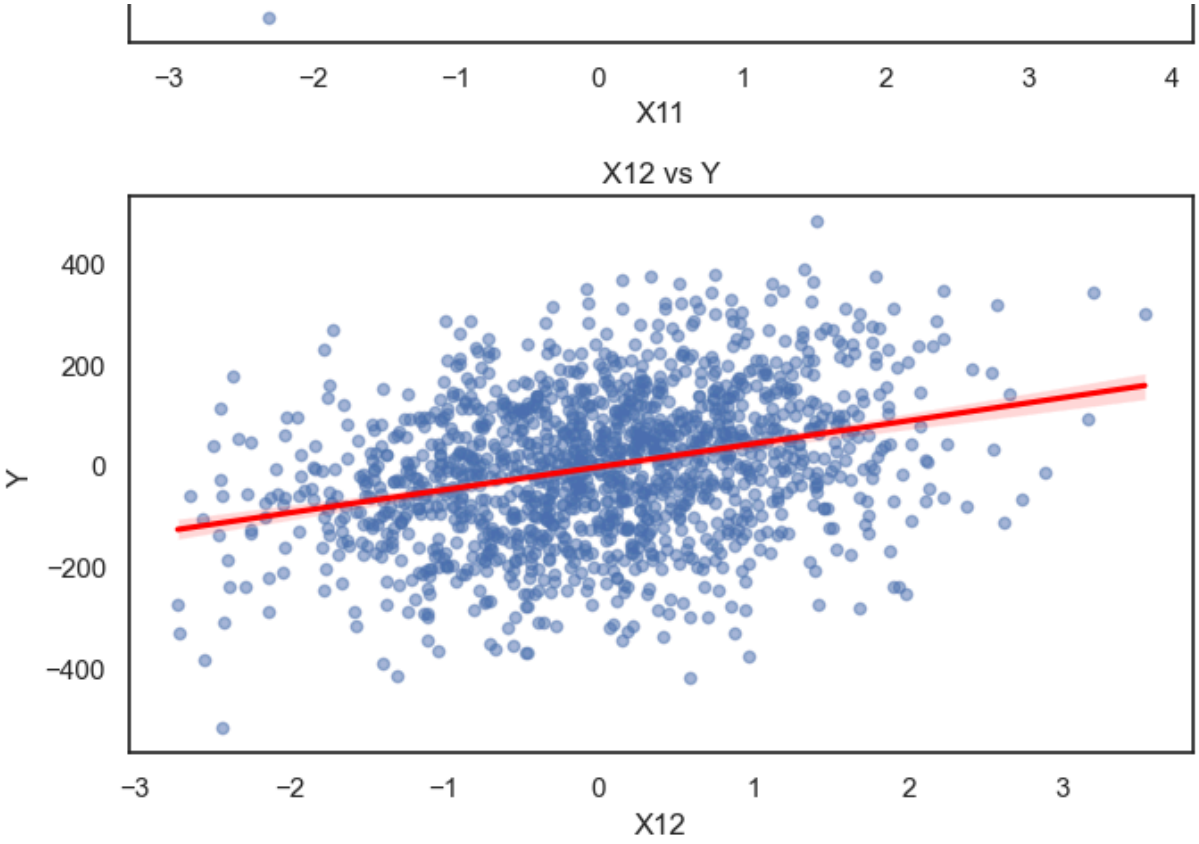

X8 vs Y
Y
400
200
0
−200
−400
−4
−3
−2
−1
0
1
2
3
X8


X9 vs Y


400
200
0
−200
−400
Y
−750000
−500000
−250000
0
250000
500000
750000
X9

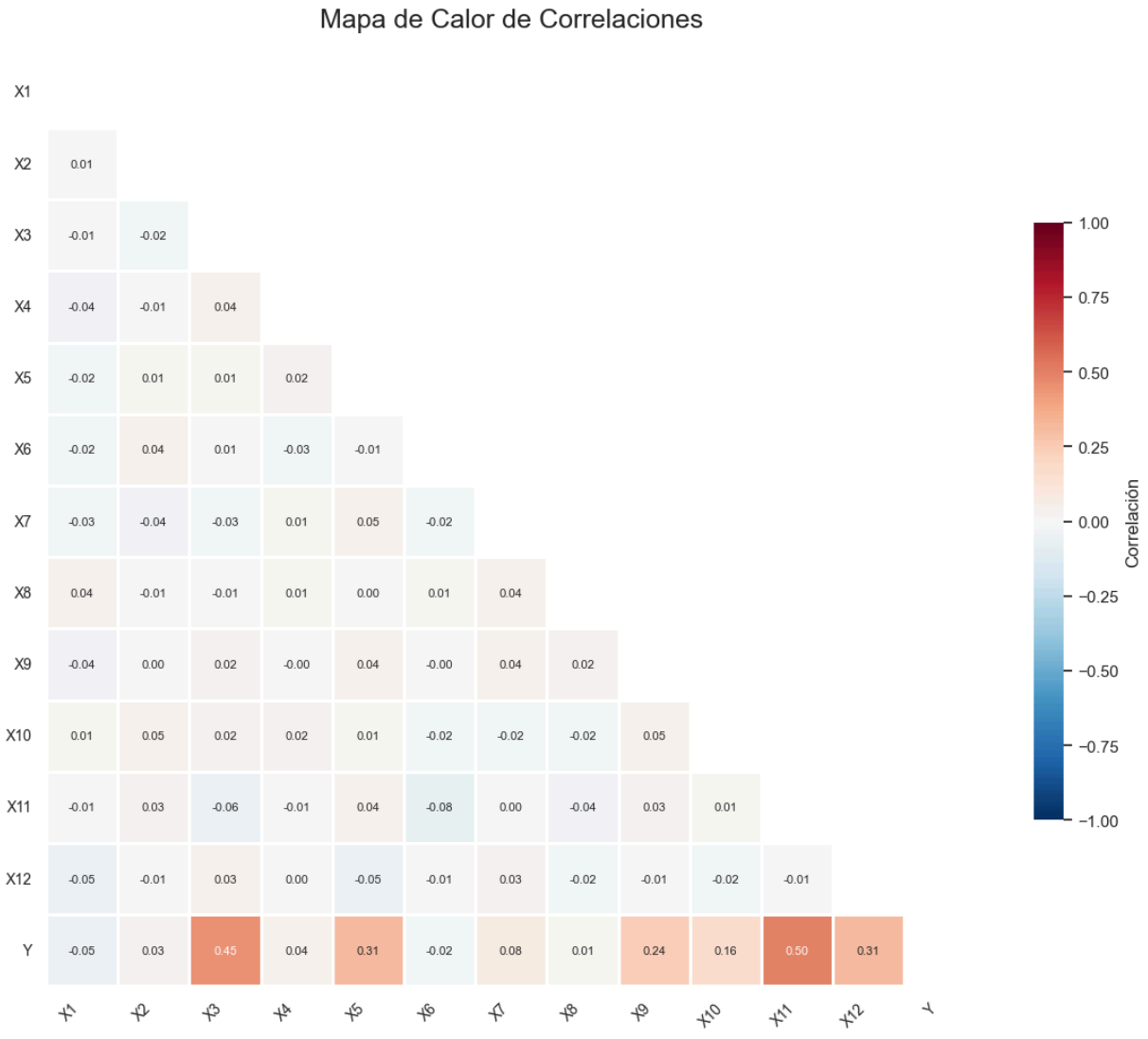

X10 vs Y
400
200
0
−200
−400
Y
−3
−2
−1
0
1
2
3
X10


X11 vs Y
400
200
0
−200
−400
Y



In [28]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

corr = df_reg[numeric_cols_reg.tolist() + ["Y"]].corr()


sns.set_theme(style="white")

mask = np.triu(np.ones_like(corr, dtype=bool))

n = corr.shape[0]
fig, ax = plt.subplots(figsize=(1.2 * n, 1.2 * n))

sns.heatmap(
    corr,
    mask=mask,
    annot=True,
    fmt=".2f",
    cmap="RdBu_r",
    center=0,
    linewidths=1.0,
    square=True,
    annot_kws={"size": 8},
    cbar_kws={"shrink": 0.6, "label": "Correlación"},
    vmin=-1,
    vmax=1,
    ax=ax,
)
```

```
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha="right", fontsize=1
ax.set_yticklabels(ax.get_yticklabels(), rotation=0, fontsize=10)

plt.subplots_adjust(bottom=0.3, left=0.2, top=0.9)

ax.set_title("Mapa de Calor de Correlaciones", fontsize=18, pad=20)

plt.show()
```



Comments

Classification

- El dataset tiene: 1.500 observaciones y 16 columnas (15 numéricas y 1 categórica de clase Y). Las variables X1–X15 son float y Y es int con valores {0,1,2}
- Menos del 1% de datos faltantes en 8 de las 15 variables (máx. 3 filas en X2/X4), representando sólo ~15 muestras; tras eliminar nulos quedan 1,485 filas.

- Percentiles 25, 50 y 75 en 0,1,2 nos indican clases equilibradas (~500 ejemplos por clase).
- Medias centradas cerca de cero, desviaciones estándar en rango 3–8.
- Casi todas las Xi presentan tres picos (formas multimodales) bien definidos en sus histogramas, lo que podrían ser agrupamientos alineados con las tres clases.
- Variables como X1, X13 o X14 muestran 3 grupos marcados.
- Los scatterplots de Y vs X1, X8, X13 nos confirman agrupamientos clasificables por rangos de cada variable.
- En el pairplot, la mayoría de pares de variables muestran tres nubes separadas.
- En el mapa de calor de correlaciones se aprecian múltiples pares con valores muy fuertes $|\rho| > 0.9$. La variable Y se correlaciona positivamente con X6 (+0.96), X7 (+0.93) y negativamente con X13 (–0.98).
- **Multicolinealidad:** Alta redundancia entre características (varias parejas >0.9), se podrían usar técnicas de selección o reducción de dimensión para evitar sobreajuste.

Regression

- El dataset contiene 1 473 observaciones y 13 columnas — 12 predictoras (X1-X12, float) y la variable objetivo Y (float).
- Menos del 2% de datos faltantes repartidos en varias Xi; tras eliminar nulos se conservan las 1 473 filas (pérdida ≈ 1,8%).
- Varias de las variables parecieran seguir distribuciones normales centradas en 0.
- X1, X3, X9 presentan rangos mucho más grandes y colas pesadas.
- Desviaciones estándar distintas y distantes en X1 (~ 12 665), X3 (~ 95 364) y X9 (~ 233 212) frente a ≈ 1 en el resto, podría considerarse escalado para aquellos modelos en los que impacte esto.
- No existe correlación con Y, la mayoría de Xi muestra $\rho \approx 0$.
- Los puntos del scatter plot no siguen un patrón claro y están muy dispersos alrededor de la línea roja (recta de regresión).
- **Multicolinealidad mínima**: coeficientes Xi-Xj muy bajos (|ρ| < 0,06), por lo que las variables son prácticamente independientes.

In [ ]: