Statistical learning and data analysis - Exercise 1 (Basic analysis of a database)

In [1]:
```python
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
import ssl
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')

import warnings
warnings.filterwarnings("ignore")
import sys
sys.stderr = sys.stdout

plt.rcParams['figure.figsize'] = (20,10)

!pip install corner
import corner
```

Part 1 - Explore our chosen data

First, loading the data, the Airbnb data we found provides a detailed analysis of Airbnb prices in popular European cities based on various features such as room types, cleanliness ratings, distance from the city center, etc.

we chose 3 europian cities (amsterdam, barcelona & berlin) for each we have data for Airbnb features that can effect the price, which is the reference variable (Y) we would like to explain.

Each city have a seperate weekdays and weekend data, we merge those observations and made a binary variable for weekend (0/1).

Then, we add another binary variable for each city to check the affect of the city on the price.

In [ ]:
```python
amst_weekdays=pd.read_csv('/content/drive/MyDrive/amsterdam_weekdays.csv')
amst_weekends=pd.read_csv('/content/drive/MyDrive/amsterdam_weekends.csv')
amst_weekdays["weekend"]=np.zeros(len(amst_weekdays))
amst_weekends["weekend"]=np.ones(len(amst_weekends))
amsterdam=pd.concat([amst_weekdays,amst_weekends])

barc_weekdays=pd.read_csv('/content/drive/MyDrive/barcelona_weekdays.csv')
barc_weekends=pd.read_csv('/content/drive/MyDrive/barcelona_weekends.csv')
barc_weekdays["weekend"]=np.zeros(len(barc_weekdays))
barc_weekends["weekend"]=np.ones(len(barc_weekends))
barcelona=pd.concat([barc_weekdays,barc_weekends])

berl_weekdays=pd.read_csv('/content/drive/MyDrive/berlin_weekdays.csv')
berl_weekends=pd.read_csv('/content/drive/MyDrive/berlin_weekends.csv')
berl_weekdays["weekend"]=np.zeros(len(berl_weekdays))
berl_weekends["weekend"]=np.ones(len(berl_weekends))
berlin=pd.concat([berl_weekdays,berl_weekends])

berlin["amsterdam"]=np.zeros(len(berlin))
berlin["barcelona"]=np.zeros(len(berlin))
berlin["berlin"]=np.ones(len(berlin))

amsterdam["amsterdam"]=np.ones(len(amsterdam))
amsterdam["barcelona"]=np.zeros(len(amsterdam))
amsterdam["berlin"]=np.zeros(len(amsterdam))

barcelona["amsterdam"]=np.zeros(len(barcelona))
barcelona["barcelona"]=np.ones(len(barcelona))
barcelona["berlin"]=np.zeros(len(barcelona))
```

Now, we merge those 3 cities data to one DataFrame for our analysis.

Also, convert another category variable for room type to 3 binary variables: private_room, Entire_home_apt & shared room.

In [ ]:
```python
abb = pd.concat([amsterdam,berlin,barcelona],ignore_index=True).iloc[:,1:]

abb["private_room"] = np.where(abb["room_type"]=="Private room",1,0)
abb["Entire_home_apt"] = np.where(abb["room_type"]=="Entire home/apt",1,0)
abb["shared_room"] = np.where(abb["room_type"]=="Shared room",1,0)

abb = abb.loc[:,abb.columns!="room_type"]
abb = abb.drop(["room_shared", "room_private", "biz", "multi", "attr_index", "attr_index_norm", "rest_index", "rest_index_norm", "lng", "lat"], 1)
abb["host_is_superhost"] = abb["host_is_superhost"].astype(int)

print("glimpse to our data: ")
abb.head()
```

glimpse to our data:

Out[ ]:

	realSum	person_capacity	host_is_superhost	cleanliness_rating	guest_satisfaction_overall	bedrooms	dist	metro_dist	weekend	amsterdam	barcelona	berlin	private_room	E
0	194.033698	2.0	0	10.0	93.0	1	5.022964	2.539380	0.0	1.0	0.0	0.0	1	
1	344.245776	4.0	0	8.0	85.0	1	0.488389	0.239404	0.0	1.0	0.0	0.0	1	
2	264.101422	2.0	0	9.0	87.0	1	5.748312	3.651621	0.0	1.0	0.0	0.0	1	
3	433.529398	4.0	0	9.0	90.0	2	0.384862	0.439876	0.0	1.0	0.0	0.0	1	
4	485.552926	2.0	1	10.0	98.0	1	0.544738	0.318693	0.0	1.0	0.0	0.0	1	

Y variable (The reference variable)

---

realSum : The total price of the Airbnb listing. (Numeric)

X variables (The explain variables)

---

shared_room : Whether the room is shared or not. (Binary)

private_room : Whether the room is private or not. (Binary)

Entire_home_apt : Whether the offer is for the entire apartment or not. (Binary)

person_capacity : The maximum number of people that can stay in the room. (Numeric)

host_is_superhost : Whether the host is a superhost or not. (Binary)

weekend : Whether the price is for weekend or not. (Binary)

cleanliness_rating : The cleanliness rating of the listing. (Numeric)

guest_satisfaction_overall : The overall guest satisfaction rating of the listing. (Numeric)

bedrooms : The number of bedrooms in the listing. (Numeric)

dist : The distance from the city centre. (Numeric)

metro_dist : The distance from the nearest metro station. (Numeric)

amsterdam : Whether the city is amsterdam or not. (Binary)

berlin : Whether the city is berlin or not. (Binary)

barcelona : Whether the city is barcelona or not. (Binary)

Some of our variables are extraneous events, meaning when one happens the other cannot exist and hence they have dependencies, if one happens it surely affects the occurrence of the other. For example room type, city.

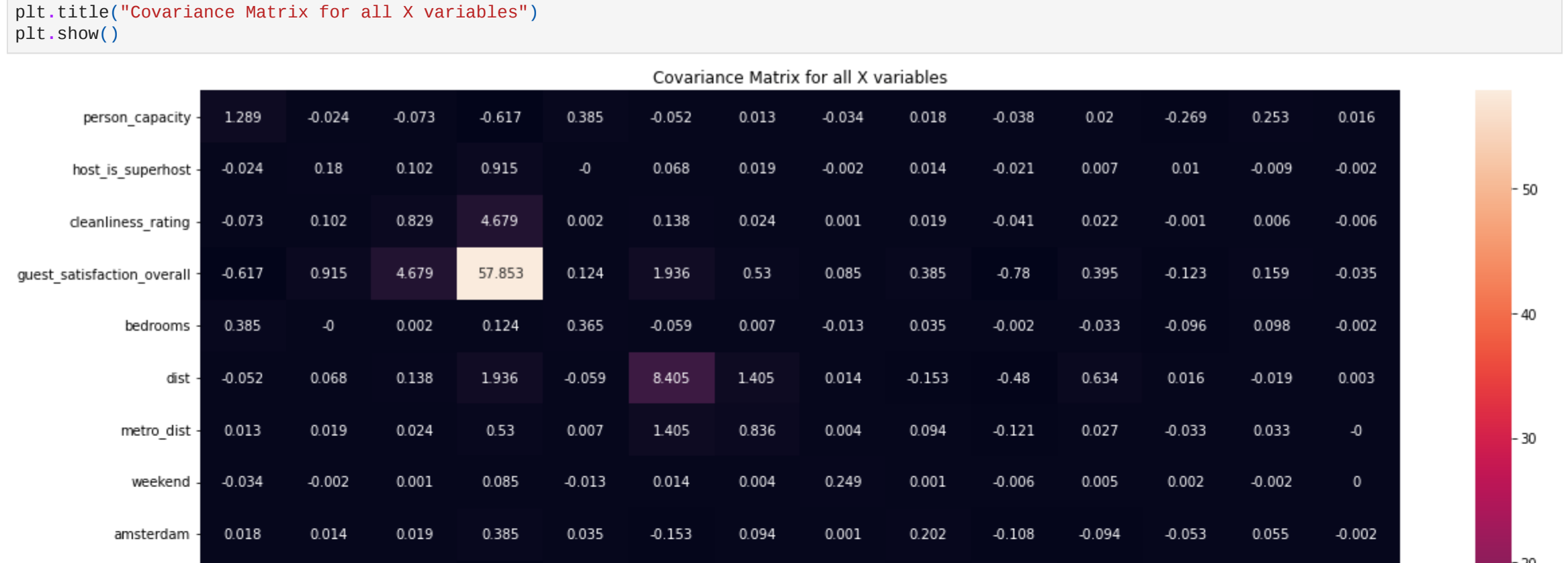
So, we expect a certain dependence between those variables.

Variables which we expect more dependence are cleanliness rating & distance from center with guest satisfaction And also distance from the city center and distance from the metro.

These variables are more influenced by each other. cleanliness and being close to the city center goes hand in hand with satisfaction and proximity to the city center and the metro will usually come together.

To more formally check for independency we take a look a the Covariance matrix of our X vars.

In [ ]:
```python
exp_vars = list(abb.columns)[1:]
cov_mat = abb[exp_vars].cov().round(3)
sns.heatmap(cov_mat, annot=True, fmt='g')
plt.title("Covariance Matrix for all X variables")
plt.show()
```

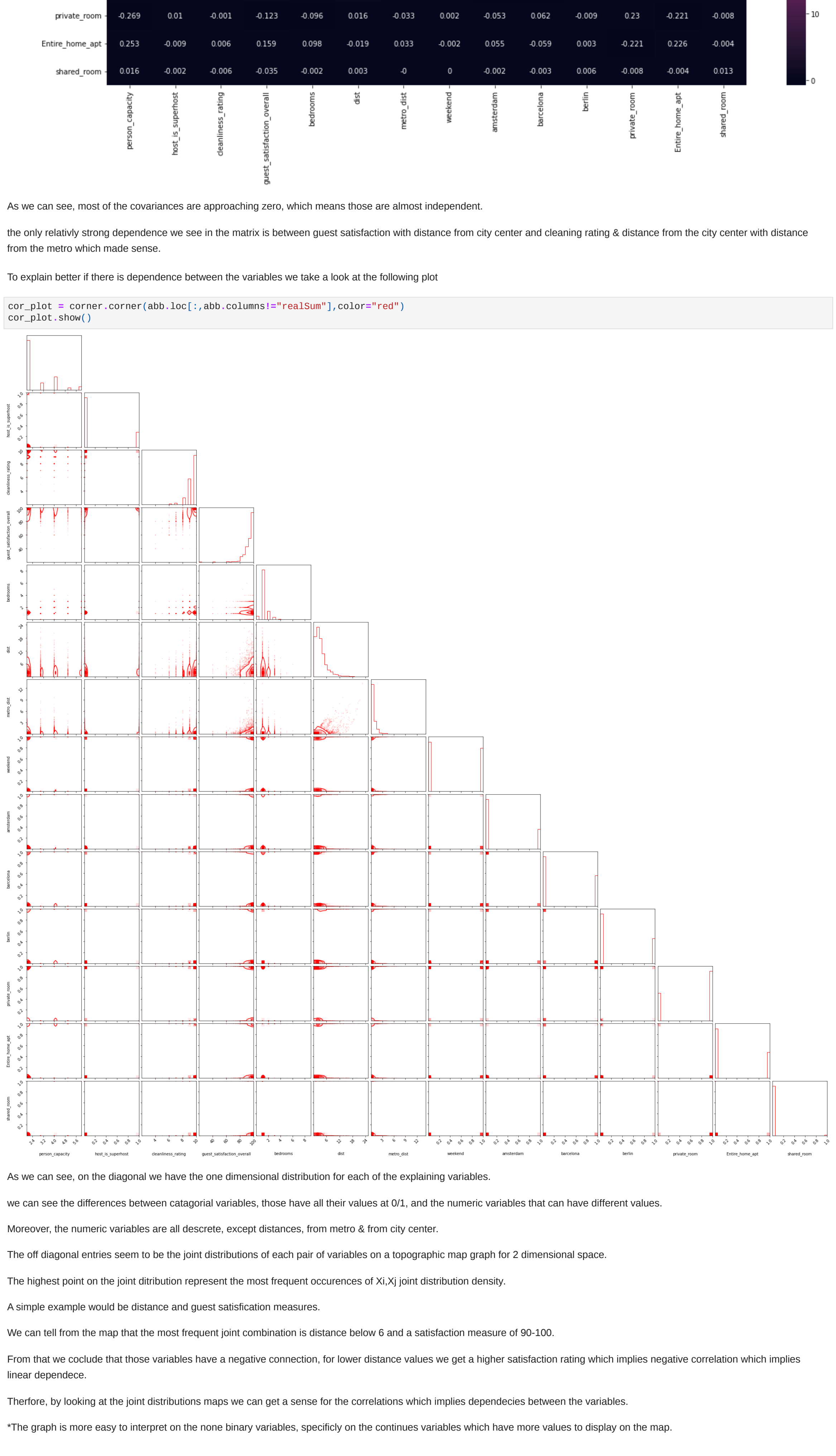


As we can see, most of the covariances are approaching zero, which means those are almost independent.

the only relativly strong dependence we see in the matrix is between guest satisfaction with distance from city center and cleaning rating & distance from the city center with distance from the metro which made sense.

To explain better if there is dependence between the variables we take a look at the following plot

In [ ]:
```python
cor_plot = corner.corner(abb.loc[:,abb.columns!="realSum"],color="red")
cor_plot.show()
```



As we can see, on the diagonal we have the one dimensional distribution for each of the explaining variables.

we can see the differences between catagorial variables, those have all their values at 0/1, and the numeric variables that can have different values.

Moreover, the numeric variables are all descrete, except distances, from metro & from city center.

The off diagonal entries seem to be the joint distributions of each pair of variables on a topographic map graph for 2 dimensional space.

The highest point on the joint ditribution represent the most frequent occurences of Xi,Xj joint distribution density.

A simple example would be distance and guest satisfication measures.

We can tell from the map that the most frequent joint combination is distance below 6 and a satisfaction measure of 90-100.

From that we coclude that those variables have a negative connection, for lower distance values we get a higher satisfaction rating which implies negative correlation which implies linear dependece.

Therfore, by looking at the joint distributions maps we can get a sense for the correlations which implies dependecies between the variables.

*The graph is more easy to interpret on the none binary variables, specificly on the continues variables which have more values to display on the map.

Part 2 - Given dataset

Linear Regression on the given dataset

In [ ]:
```python
df = pd.read_csv('/content/drive/MyDrive/dataset.csv')
cols = list(df.columns)
df.head() # glimpse to the data
```

Out[ ]:

	y	x1	x2	x3	x4	x5	x6	x7	x8	x9	...	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24
0	107	102.886643	196.943843	297.906121	52	22	32.859792	164	106	63.507651	...	125.475051	0	1	0	154.783184	147.622883	244.667677	0.369351	0.500646	1.475186
1	90	100.815798	194.741136	300.260251	49	19	17.459270	161	96	70.442053	...	88.476607	0	1	0	50.996887	84.631845	113.862044	0.637361	0.511120	0.640851
2	90	105.404634	199.422587	301.454232	46	13	26.533067	150	98	99.324123	...	113.794568	1	1	1	24.581021	43.355957	80.561859	0.557686	0.522909	-0.015851
3	92	109.501591	206.921739	317.415536	41	13	26.516413	152	109	96.347172	...	93.474055	1	1	1	54.475980	134.528252	212.358502	0.416433	0.510081	0.465479
4	99	93.237287	195.307733	286.121573	48	13	23.520136	136	93	61.732369	...	115.317433	1	1	0	34.491754	331.263536	87.449725	0.723308	0.514442	1.992274

5 rows × 25 columns

In [ ]:
```python
x = df[cols[1:]]
x = sm.add_constant(x) # intercept vector
y = df["y"]
model = sm.OLS(y, x).fit()

print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.680
Model:                            OLS   Adj. R-squared:                  0.664
Method:                 Least Squares   F-statistic:                     42.04
Date:                Sat, 25 Mar 2023   Prob (F-statistic):          2.05e-101
Time:                        09:03:20   Log-Likelihood:                -1510.7
No. Observations:                 500   AIC:                             3071.
Df Residuals:                     475   BIC:                             3177.
Df Model:                          24
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -12.3999      8.936     -1.388      0.166     -29.958       5.158
x1            -0.0624      0.037     -1.700      0.090      -0.135       0.010
x2             0.0499      0.036      1.399      0.163      -0.020       0.120
x3             0.1575      0.025      6.290      0.000       0.108       0.207
x4             0.0615      0.036      1.724      0.085      -0.009       0.132
x5            -0.0398      0.051     -0.784      0.434      -0.140       0.060
x6             0.1566      0.023      6.730      0.000       0.111       0.202
x7             0.1016      0.034      2.970      0.003       0.034       0.169
x8             0.0830      0.035      2.397      0.017       0.015       0.151
x9             0.0014      0.016      0.087      0.931      -0.030       0.033
x10           -0.0190      0.018     -1.062      0.289      -0.054       0.016
x11            0.0982      0.023      4.254      0.000       0.053       0.144
x12            0.0485      0.024      1.988      0.047       0.001       0.096
x13           -0.0046      0.024     -0.195      0.845      -0.051       0.042
x14            0.0268      0.023      1.144      0.253      -0.019       0.073
x15           -0.0181      0.023     -0.802      0.423      -0.063       0.026
x16           -0.0580      0.528     -0.110      0.912      -1.095       0.979
x17            0.2191      0.530      0.413      0.679      -0.822       1.260
x18           -0.8126      0.808     -1.006      0.315      -2.399       0.774
x19            0.0474      0.004     10.637      0.000       0.039       0.056
x20            0.0493      0.002     22.459      0.000       0.045       0.054
x21            0.0011      0.002      0.475      0.635      -0.003       0.006
x22            0.5173      2.203      0.235      0.815      -3.812       4.847
x23           10.2715      6.954      1.477      0.140      -3.392      23.935
x24            1.2009      0.237      5.075      0.000       0.736       1.666
==============================================================================
Omnibus:                        0.790   Durbin-Watson:                   2.017
Prob(Omnibus):                  0.674   Jarque-Bera (JB):                0.881
Skew:                           0.087   Prob(JB):                        0.644
Kurtosis:                       2.890   Cond. No.                     2.18e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.18e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Now, we take a look at the P-Values of our explanatory variables

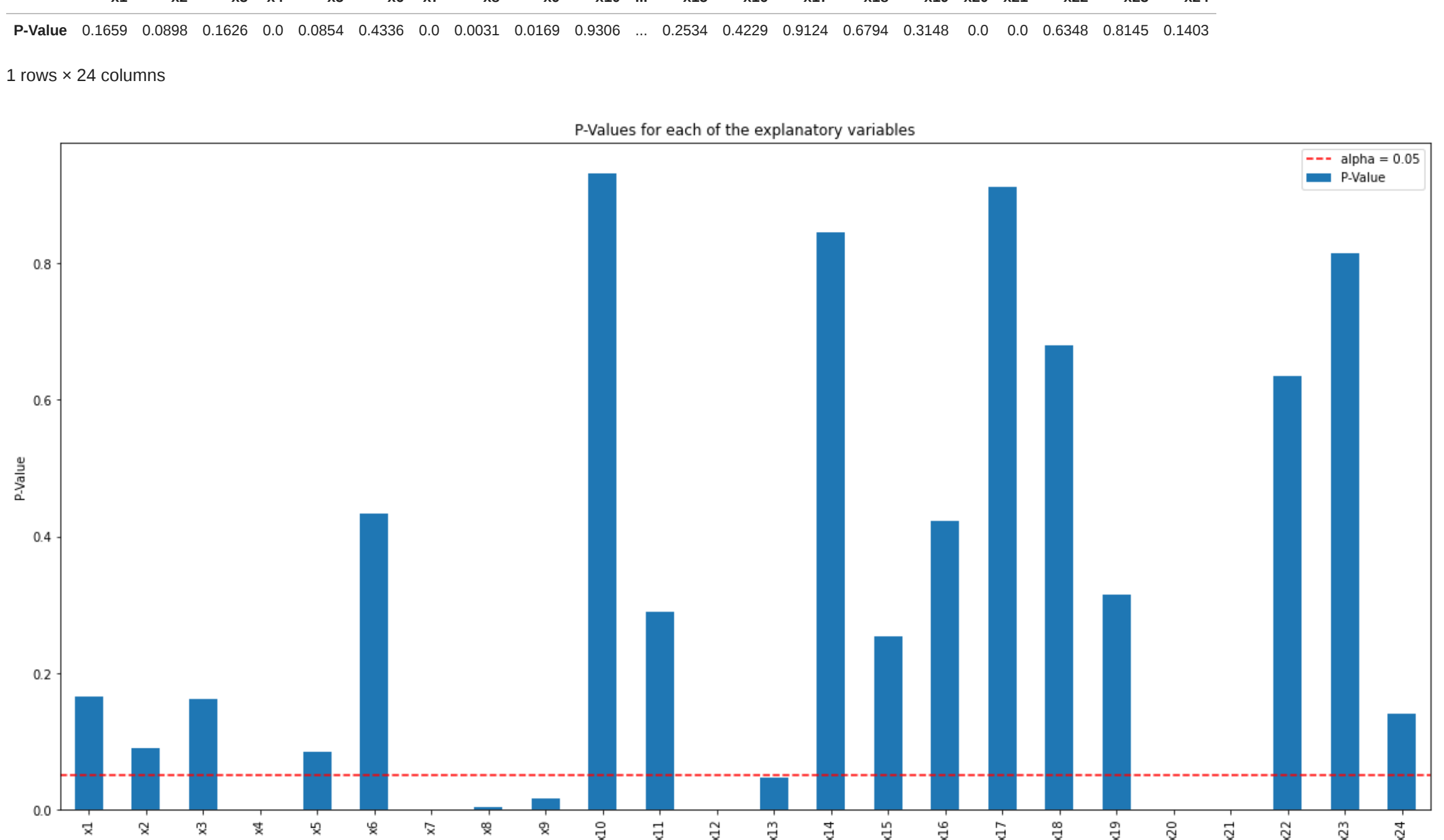
In [ ]:
```python
p = []
for x in range (0, 24):
    p.append(model.pvalues[x].round(4))
p_vals = pd.DataFrame(columns = cols[1:]).T
p_vals["P-Value"] = p

p_vals.plot.bar()
plt.axhline(y=0.05, color = "red", linestyle='dashed')
plt.legend(["alpha = 0.05", "P-Value"])
plt.xlabel("explanatory variable")
plt.ylabel("P-Value")
plt.title("P-Values for each of the explanatory variables")
p_vals.T
```

Out[ ]:

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	...	x15	x16	x17	x18	x19	x20	x21	x22	x23	x24
P-Value	0.1659	0.0898	0.1626	0.0	0.0854	0.4336	0.0	0.0031	0.0169	0.9306	...	0.2534	0.4229	0.9124	0.6794	0.3148	0.0	0.0	0.6348	0.8145	0.1403

1 rows × 24 columns



In [ ]:
```python
alpha = 0.05
x_sgnf = list(p_vals[p_vals["P-Value"] < alpha].T.columns) # sgnificant variables
print(f"The significant variables in our data are: {x_sgnf}")
```

```
The significant variables in our data are: ['x4', 'x7', 'x8', 'x9', 'x12', 'x13', 'x20', 'x21']
```

In linear regression if the p-value of the explaining variable is smaller than the significance level α it means that this variable is relevant and effect the predicted variable value. In our data, we can see 8 variables that are below the alpha line, which means that those are the relevent variables to explain best the response variable Y.

Backward elimination to our Data

In [ ]:
```python
all_x = list(p_vals.sort_values(by="P-Value").T.columns)

def get_adj_r2():
    x = df[all_x]
    x = sm.add_constant(x) # intercept vector
    y = df["y"]
    model = sm.OLS(y, x).fit()
    return model.rsquared_adj

not_relv_cols = []
for i in list(p_vals.sort_values(by="P-Value").T.columns):
  adj_r2 = get_adj_r2()
  all_x.remove(i)
  new_adj_r2 = get_adj_r2()
  if new_adj_r2 > adj_r2:
    not_relv_cols.append(i)
  elif new_adj_r2 < adj_r2:
    all_x.append(i)


relv_df = df.drop(not_relv_cols, 1)
print(f"The Backwards Elimination procces removes the columns: {not_relv_cols}\n\n Glimpse to our new Data:")
relv_df.head()
```

```
The Backwards Elimination procces removes the columns: ['x21', 'x9', 'x13', 'x5', 'x15', 'x16', 'x22', 'x17', 'x10']

 Glimpse to our new Data:
```

Out[ ]:

	y	x1	x2	x3	x4	x6	x7	x8	x11	x12	x14	x18	x19	x20	x23	x24
0	107	102.886643	196.943843	297.906121	52	32.859792	164	106	61.224838	117.639893	90.730767	0	154.783184	147.622883	0.500646	1.475186
1	90	100.815798	194.741136	300.260251	49	17.459270	161	96	36.081437	108.976700	109.565453	0	50.996887	84.631845	0.511120	0.640851
2	90	105.404634	199.422587	301.454232	46	26.533067	150	98	65.465373	81.267880	111.929653	1	24.581021	43.355957	0.522909	-0.015851
3	92	109.501591	206.921739	317.415536	41	26.516413	152	109	76.500526	89.462363	107.693666	1	54.475980	134.528252	0.510081	0.465479
4	99	93.237287	195.307733	286.121573	48	23.520136	136	93	38.801340	118.164882	115.037715	0	34.491754	331.263536	0.514442	1.992274

As we can see, the Backwards Elimination procces removes 9 variables, 2 of them are significant by their p-value.

Most of the variables we removed seem to be not sagnificant, with p-value greater then alpha = 0.05.

However, X9, X13 & X21 are sgnificant and were removed also, to explain this we can look at a scatter plot of all the removed variables agains Y to understand better the connection that explain the removal of those columns.

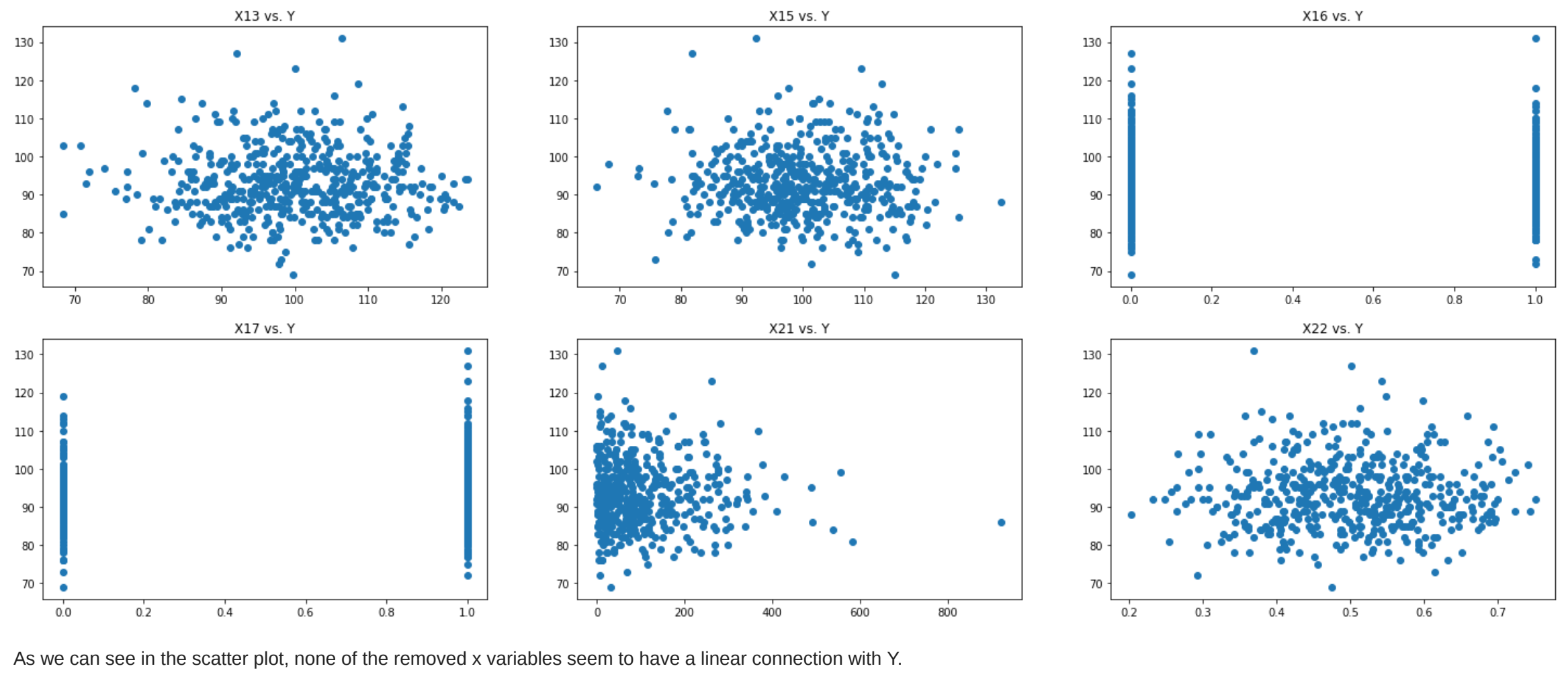
We plot each of the removed X variables against Y

In [ ]:
```python
fig, axs = plt.subplots(nrows=3, ncols=3, figsize=(25, 15))

axs[0, 0].scatter(df["x5"], y)
axs[0, 1].scatter(df["x9"], y)
axs[0, 2].scatter(df["x10"], y)
axs[1, 0].scatter(df["x13"], y)
axs[1, 1].scatter(df["x15"], y)
axs[1, 2].scatter(df["x16"], y)
axs[2, 0].scatter(df["x17"], y)
axs[2, 1].scatter(df["x21"], y)
axs[2, 2].scatter(df["x22"], y)

axs[0, 0].set(title='X5 vs. Y')
axs[0, 1].set(title='X9 vs. Y')
axs[0, 2].set(title='X10 vs. Y')
axs[1, 0].set(title='X13 vs. Y')
axs[1, 1].set(title='X15 vs. Y')
axs[1, 2].set(title='X16 vs. Y')
axs[2, 0].set(title='X17 vs. Y')
axs[2, 1].set(title='X21 vs. Y')
axs[2, 2].set(title='X22 vs. Y')

plt.show()
```



As we can see in the scatter plot, none of the removed x variables seem to have a linear connection with Y.

X9, X13 & 21 which were significant by p-value, does not have a strong connection with the response variable.

Therefore, those variables relation to the response variable (Y) cannot explain it in a good order, that can explain their removal in the Backward Elimination procces.