In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn import linear_model as lm
```

In [4]:

```
x_1 = np.random.uniform(10,40,100)
x_2 = np.random.uniform(-50,20,100)
x_3 = np.random.uniform(20,60,100)
x_4 = np.random.uniform(10,40,100)
x_5 = np.random.uniform(-50,20,100)
x_6 = np.random.uniform(20,60,100)
epsilon = np.random.normal(0,10,100)
```

In [5]:

```
y=-30+1.3*x_1+1.6*x_2+1.1*x_3+0.7*x_4-2.1*x_5-0.9*x_6+epsilon
```

In [9]:

```
X_ols=pd.DataFrame()
X_ols['Constant']=pd.Series(np.ones(100))
X_ols['X1'] = pd.Series(x_1)
X_ols['X2'] = pd.Series(x_2)
X_ols['X3'] = pd.Series(x_3)
X_ols['X4'] = pd.Series(x_4)
X_ols['X5'] = pd.Series(x_5)
X_ols['X6'] = pd.Series(x_6)
```

In [10]:

```
model_reg = sm.OLS(y,X_ols).fit()
model_reg.summary()
```

Out[10]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.967
Model:	OLS	**Adj. R-squared:**	0.965
Method:	Least Squares	**F-statistic:**	454.0
Date:	Sun, 02 Oct 2022	**Prob (F-statistic):**	1.42e-66
Time:	14:07:07	**Log-Likelihood:**	-370.82
No. Observations:	100	**AIC:**	755.6
Df Residuals:	93	**BIC:**	773.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Constant	-29.3296	6.731	-4.357	0.000	-42.696	-15.963
X1	1.2386	0.118	10.526	0.000	1.005	1.472
X2	1.5405	0.053	29.295	0.000	1.436	1.645

X3	1.0986	0.090	12.203	0.000	0.920	1.277
X4	0.7359	0.118	6.229	0.000	0.501	0.971
X5	-2.1346	0.054	-39.287	0.000	-2.242	-2.027
X6	-0.9411	0.090	-10.503	0.000	-1.119	-0.763

Omnibus:	1.259	**Durbin-Watson:**	1.845
Prob(Omnibus):	0.533	**Jarque-Bera (JB):**	1.122
Skew:	-0.074	**Prob(JB):**	0.571
Kurtosis:	2.502	**Cond. No.**	466.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [11]:

```
# Now we add some unrelated coefficients
x_7 = np.random.uniform(10,40,100)
x_8 = np.random.uniform(-50,20,100)
x_9 = np.random.uniform(20,60,100)
x_10 = np.random.uniform(10,40,100)
x_11 = np.random.uniform(-50,20,100)
x_12 = np.random.uniform(20,60,100)
```

In [23]:

```
X_ext = pd.DataFrame()
X_ext['Constant']=pd.Series(np.ones(100))
X_ext['X1'] = pd.Series(x_1)
X_ext['X2'] = pd.Series(x_2)
X_ext['X3'] = pd.Series(x_3)
X_ext['X4'] = pd.Series(x_4)
X_ext['X5'] = pd.Series(x_5)
X_ext['X6'] = pd.Series(x_6)
X_ext['X7'] = pd.Series(x_7)
X_ext['X8'] = pd.Series(x_8)
X_ext['X9'] = pd.Series(x_9)
X_ext['X10'] = pd.Series(x_10)
X_ext['X11'] = pd.Series(x_11)
X_ext['X12'] = pd.Series(x_12)
```

In [24]:

```
model_reg = sm.OLS(y,X_ext).fit()
model_reg.summary()
```

Out[24]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.968
Model:	OLS	**Adj. R-squared:**	0.963
Method:	Least Squares	**F-statistic:**	216.9
Date:	Sun, 02 Oct 2022	**Prob (F-statistic):**	2.30e-59

Time:	14:26:02	**Log-Likelihood:**	-369.80
No. Observations:	100	**AIC:**	765.6
Df Residuals:	87	**BIC:**	799.5
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Constant	-33.6749	10.495	-3.209	0.002	-54.535	-12.815
X1	1.2531	0.125	10.034	0.000	1.005	1.501
X2	1.5338	0.057	27.035	0.000	1.421	1.647
X3	1.1104	0.095	11.655	0.000	0.921	1.300
X4	0.7032	0.124	5.651	0.000	0.456	0.951
X5	-2.1466	0.057	-37.779	0.000	-2.260	-2.034
X6	-0.9398	0.095	-9.872	0.000	-1.129	-0.751
X7	0.1203	0.142	0.849	0.398	-0.161	0.402
X8	-0.0193	0.057	-0.342	0.733	-0.132	0.093
X9	-0.0273	0.098	-0.279	0.781	-0.222	0.167
X10	-0.0554	0.132	-0.420	0.676	-0.318	0.207
X11	-0.0423	0.058	-0.733	0.465	-0.157	0.072
X12	0.0625	0.098	0.638	0.525	-0.132	0.257

Omnibus:	0.892	**Durbin-Watson:**	1.793
Prob(Omnibus):	0.640	**Jarque-Bera (JB):**	0.901
Skew:	-0.056	**Prob(JB):**	0.637
Kurtosis:	2.549	**Cond. No.**	995.

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [28]:

```
# When we input the matrix, we don't need to add the column of ones because \
# Lasso automatically takes it into account

X_ext = X_ext.drop(columns=['Constant'])
model_lasso = lm.Lasso(alpha=1).fit(X_ext,y)
model_lasso.coef_
```

Out[28]:

```
array([ 1.23952752,  1.53145407,  1.09882225,  0.69359863, -2.14047446,
       -0.92871513,  0.10059897, -0.01586398, -0.01655489, -0.03638864,
       -0.03742941,  0.05533014])
```

In [36]:
```
model_lasso = lm.Lasso(alpha=10).fit(X_ext,y)
model_lasso.coef_
```

Out[36]:
```
array([ 1.10484424,  1.51390299,  1.01218205,  0.59247201, -2.09748408,
       -0.84828585,  0.        , -0.        , -0.        , -0.        ,
       -0.        ,  0.        ])
```

In [41]:
```
# We now apply Ridge Regression to our data

model_ridge = lm.Ridge(alpha=10).fit(X_ext,y)
model_ridge.coef_
```

Out[41]:
```
array([ 1.25127262,  1.53337585,  1.10937072,  0.70229948, -2.14587799,
       -0.93878662,  0.11997347, -0.01943572, -0.0270754 , -0.05503608,
       -0.04212088,  0.06266356])
```

In [42]:
```
model_ridge = lm.Ridge(alpha=10000).fit(X_ext,y)
model_ridge.coef_
```

Out[42]:
```
array([ 0.50321658,  1.21756962,  0.57935289,  0.32105205, -1.62079055,
       -0.42170529,  0.00375219, -0.07646285,  0.07164929,  0.05336775,
        0.01281994,  0.08499685])
```

In [ ]: