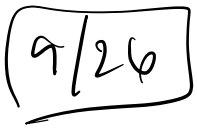
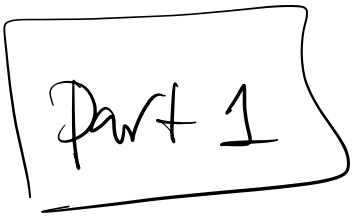




Exercise

For the dataset "multicollinearity.txt", perform the following:

1. Check the pairwise correlations of the predictors.
2. Compare the coefficient for each predictor in two cases: a. when regressing Y on that predictor only, and b. when regressing Y on all predictors. Do you see a major change in the coefficients? Their standard errors? What does this tell you about the presence of multicollinearity?
3. Calculate the VIFs for each predictor. Which VIFs are concerning? Comment on what you find.
4. Consider regressing `X4` onto the other predictors `X1` through `X3`. Save the residuals from this model and call them `res4`. Then regress Y onto `res4` and look at the coefficient. Compare this coefficient to the coefficients for `X4` you found in part 2. What do you notice?
5. Explain geometrically what we are doing in #4. Are we surprised by the results?

In [34]:
```
data = pd.read_csv("~/Desktop/repos/regression_f24/data/multicollinearity.tx
```

In [35]:
```
data.head()
```

Out[35]:

	y	x1	x2	x3	x4
0	1.860	0.374	0.846	1.951	1.325
1	9.747	2.184	3.754	5.099	5.372
2	12.707	2.164	3.635	2.505	2.582
3	12.822	5.595	4.805	3.653	3.399
4	5.160	5.330	2.611	1.873	1.474

In [36]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   y       200 non-null    float64
 1   x1      200 non-null    float64
 2   x2      200 non-null    float64
 3   x3      200 non-null    float64
 4   x4      200 non-null    float64
dtypes: float64(5)
memory usage: 7.9 KB
```

In [37]:
```
data.iloc[:, 1:].corr()
```

Out[37]:

	x1	x2	x3	x4
x1	1.000000	0.999811	0.999495	0.998843
x2	0.999811	1.000000	0.999671	0.999013
x3	0.999495	0.999671	1.000000	0.999306
x4	0.998843	0.999013	0.999306	1.000000

The coefficient for X1 goes from being pistive at 2.25 to being negative at -0.1675 when regressed alongside all predictors. The std error grow by a factor of ~ 50x when other correlated predictors are added to the regression alongside x1.

In [39]:
```
mod_y_x1 = ols("y ~ x1", data=data).fit()
mod_y_x1.summary()
```

Out[39]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.396e+05
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	5.39e-284
Time:	22:02:42	**Log-Likelihood:**	-589.73
No. Observations:	200	**AIC:**	1183.
Df Residuals:	198	**BIC:**	1190.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.7257	0.659	2.617	0.010	0.426	3.026
x1	2.1250	0.006	373.575	0.000	2.114	2.136

Omnibus:	0.666	**Durbin-Watson:**	2.009
Prob(Omnibus):	0.717	**Jarque-Bera (JB):**	0.451
Skew:	-0.106	**Prob(JB):**	0.798
Kurtosis:	3.095	**Cond. No.**	233.

Notes:

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

In [40]:

```
mod_y_all = ols("y ~ x1 + x2 + x3 + x4", data=data).fit()
mod_y_all.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	7.200e+04
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	8.92e-308
Time:	22:02:42	**Log-Likelihood:**	-515.82
No. Observations:	200	**AIC:**	1042.
Df Residuals:	195	**BIC:**	1058.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0374	0.462	2.248	0.026	0.127	1.948
x1	-0.1675	0.204	-0.821	0.413	-0.570	0.235
x2	1.1364	0.281	4.050	0.000	0.583	1.690
x3	0.9788	0.256	3.830	0.000	0.475	1.483
x4	1.1175	0.210	5.311	0.000	0.703	1.533

Omnibus:	0.460	**Durbin-Watson:**	2.103
Prob(Omnibus):	0.795	**Jarque-Bera (JB):**	0.219
Skew:	-0.046	**Prob(JB):**	0.896
Kurtosis:	3.133	**Cond. No.**	377.

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

The coefficient for X2 changes but stays positive. The standard error increases by a factor of 60 times (!). This is variance inflation!

In [42]:
```
mod_y_x2 = ols("y ~ x2", data=data).fit()
mod_y_x2.summary()
```

Out[42]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.980e+05
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	5.20e-299
Time:	22:02:42	**Log-Likelihood:**	-554.80
No. Observations:	200	**AIC:**	1114.
Df Residuals:	198	**BIC:**	1120.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.3706	0.554	2.473	0.014	0.278	2.464
x2	2.3613	0.005	444.944	0.000	2.351	2.372

Omnibus:	0.045	**Durbin-Watson:**	1.978
Prob(Omnibus):	0.978	**Jarque-Bera (JB):**	0.043
Skew:	0.030	**Prob(JB):**	0.979
Kurtosis:	2.960	**Cond. No.**	210.

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

The coefficient changes, but sign stays the same. The SE grows by a factor of 50x...!

In [44]:
```
mod_y_x3 = ols("y ~ x3", data=data).fit()
mod_y_x3.summary()
```

Out[44]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	2.218e+05
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	6.86e-304
Time:	22:02:42	**Log-Likelihood:**	-543.45
No. Observations:	200	**AIC:**	1091.
Df Residuals:	198	**BIC:**	1098.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0831	0.524	2.066	0.040	0.049	2.117
x3	2.9492	0.006	470.945	0.000	2.937	2.962

Omnibus:	0.792	**Durbin-Watson:**	2.128
Prob(Omnibus):	0.673	**Jarque-Bera (JB):**	0.741
Skew:	-0.148	**Prob(JB):**	0.690
Kurtosis:	2.965	**Cond. No.**	169.

Notes:

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

In [45]:

```
m_y_x4 = ols("y ~ x4", data=data).fit()
m_y_x4.summary()
```

Out[45]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.998
Model:	OLS	**Adj. R-squared:**	0.998
Method:	Least Squares	**F-statistic:**	1.307e+05
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	3.46e-281
Time:	22:02:42	**Log-Likelihood:**	-596.26
No. Observations:	200	**AIC:**	1197.
Df Residuals:	198	**BIC:**	1203.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.8964	0.683	1.312	0.191	-0.451	2.244
x4	4.2002	0.012	361.558	0.000	4.177	4.223

Omnibus:	0.500	**Durbin-Watson:**	2.234
Prob(Omnibus):	0.779	**Jarque-Bera (JB):**	0.641
Skew:	0.079	**Prob(JB):**	0.726
Kurtosis:	2.772	**Cond. No.**	119.

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

VIF

The VIF values for all the predictors are very high, pointing to severe multicollinearity in the model, confirming the conclusion of the previous question

In [48]:
```
def calculate_VIF(r2_j):
    return (1/(1-r2_j))
```

In [49]:
```
mdl_x1 = ols("x1~x2+x3+x4", data=data).fit()
r2j_x1 = mdl_x1.rsquared
print(r2j_x1)
print(calculate_VIF(r2j_x1))
```

```
0.9996231915946605
2653.8686128804293
```

In [50]:
```
mdl_x2 = ols("x2~x1+x3+x4", data=data).fit()
r2j_x2 = mdl_x2.rsquared
print(r2j_x2)
print(calculate_VIF(r2j_x2))
```

```
0.9997540700184278
4066.1980032159418
```

In [51]:
```
mdl_x3 = ols("x3~x2+x1+x4", data=data).fit()
r2j_x3 = mdl_x3.rsquared
print(r2j_x3)
print(calculate_VIF(r2j_x3))
```

```
0.9995375633481587
2162.4583519023327
```

In [52]:
```
mdl_x4 = ols("x4~x2+x3+x1", data=data).fit()
r2j_x4 = mdl_x4.rsquared
print(r2j_x4)
print(calculate_VIF(r2j_x4))
```

```
0.9986155124710678
722.2889185367274
```

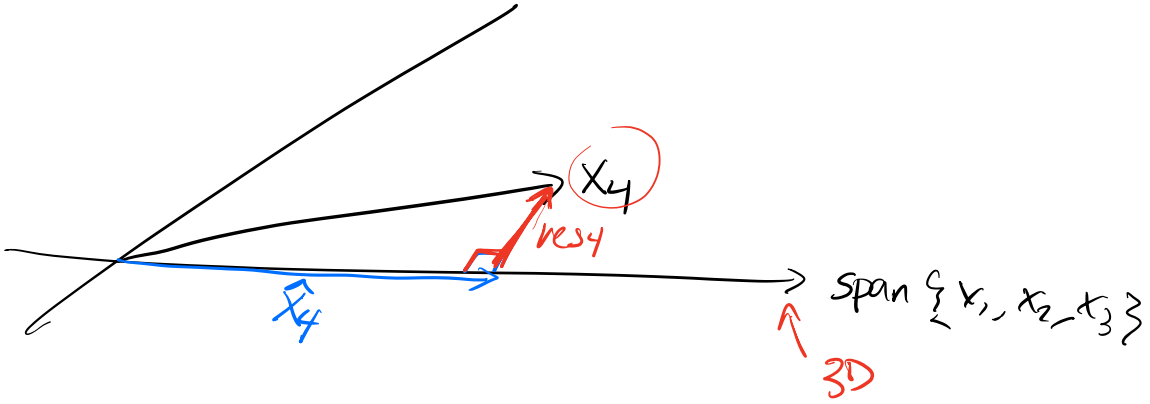
4. $res4 = X_4 - \hat{X}_4$ $\quad X_4 \sim X_1 + X_2 + X_3$

The coefficient for res4 is the same as that of X4 in the model with all the predictors and of res4 in the model with all the other predictors. (Key observation: $res4 \perp span\{X_1, X_2, X_3\}$.)

In [54]:
```
modres = ols("x4~x1+x2+x3", data=data).fit()
res4 = modres.resid
```

In [55]:
```
data['res4'] = res4
```

In [56]:
```
mod_y_res4 = ols("y~res4", data=data).fit()
mod_y_res4.summary()
```



Out[56]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.005
Method:	Least Squares	**F-statistic:**	0.01938
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	0.889
Time:	22:02:42	**Log-Likelihood:**	-1245.7
No. Observations:	200	**AIC:**	2495.
Df Residuals:	198	**BIC:**	2502.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	215.3599	8.716	24.707	0.000	198.171	232.549
res4	1.1175	8.028	0.139	0.889	-14.714	16.949

Omnibus:	91.438	**Durbin-Watson:**	0.003
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	11.967
Skew:	-0.004	**Prob(JB):**	0.00252
Kurtosis:	1.802	**Cond. No.**	1.09

Notes:

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

In [57]:

```
mod_y_x123res4 = ols("y~x1+x2+x3+res4", data=data).fit()
mod_y_x123res4.summary()
```

Out[57]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	7.200e+04
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	8.92e-308
Time:	22:02:42	**Log-Likelihood:**	-515.82
No. Observations:	200	**AIC:**	1042.
Df Residuals:	195	**BIC:**	1058.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.1484	0.461	2.491	0.014	0.239	2.058
x1	-0.1589	0.204	-0.779	0.437	-0.561	0.244
x2	1.1616	0.281	4.140	0.000	0.608	1.715
x3	1.7193	0.214	8.027	0.000	1.297	2.142
res4	1.1175	0.210	5.311	0.000	0.703	1.533

Omnibus:	0.460	**Durbin-Watson:**	2.103
Prob(Omnibus):	0.795	**Jarque-Bera (JB):**	0.219
Skew:	-0.046	**Prob(JB):**	0.896
Kurtosis:	3.133	**Cond. No.**	358.

Notes:

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

In [58]:

```
mod_y_all = ols("y ~ x1 + x2 + x3 + x4", data=data).fit()
mod_y_all.summary()
```

Out[58]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	7.200e+04
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	8.92e-308
Time:	22:02:42	**Log-Likelihood:**	-515.82
No. Observations:	200	**AIC:**	1042.
Df Residuals:	195	**BIC:**	1058.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.0374	0.462	2.248	0.026	0.127	1.948
x1	-0.1675	0.204	-0.821	0.413	-0.570	0.235
x2	1.1364	0.281	4.050	0.000	0.583	1.690
x3	0.9788	0.256	3.830	0.000	0.475	1.483
x4	1.1175	0.210	5.311	0.000	0.703	1.533

Omnibus:	0.460	**Durbin-Watson:**	2.103
Prob(Omnibus):	0.795	**Jarque-Bera (JB):**	0.219
Skew:	-0.046	**Prob(JB):**	0.896
Kurtosis:	3.133	**Cond. No.**	377.

Notes:

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

But note the coefficient is different from that in the model with just y onto x4:

In [60]:
```
m_y_x4.summary()
```

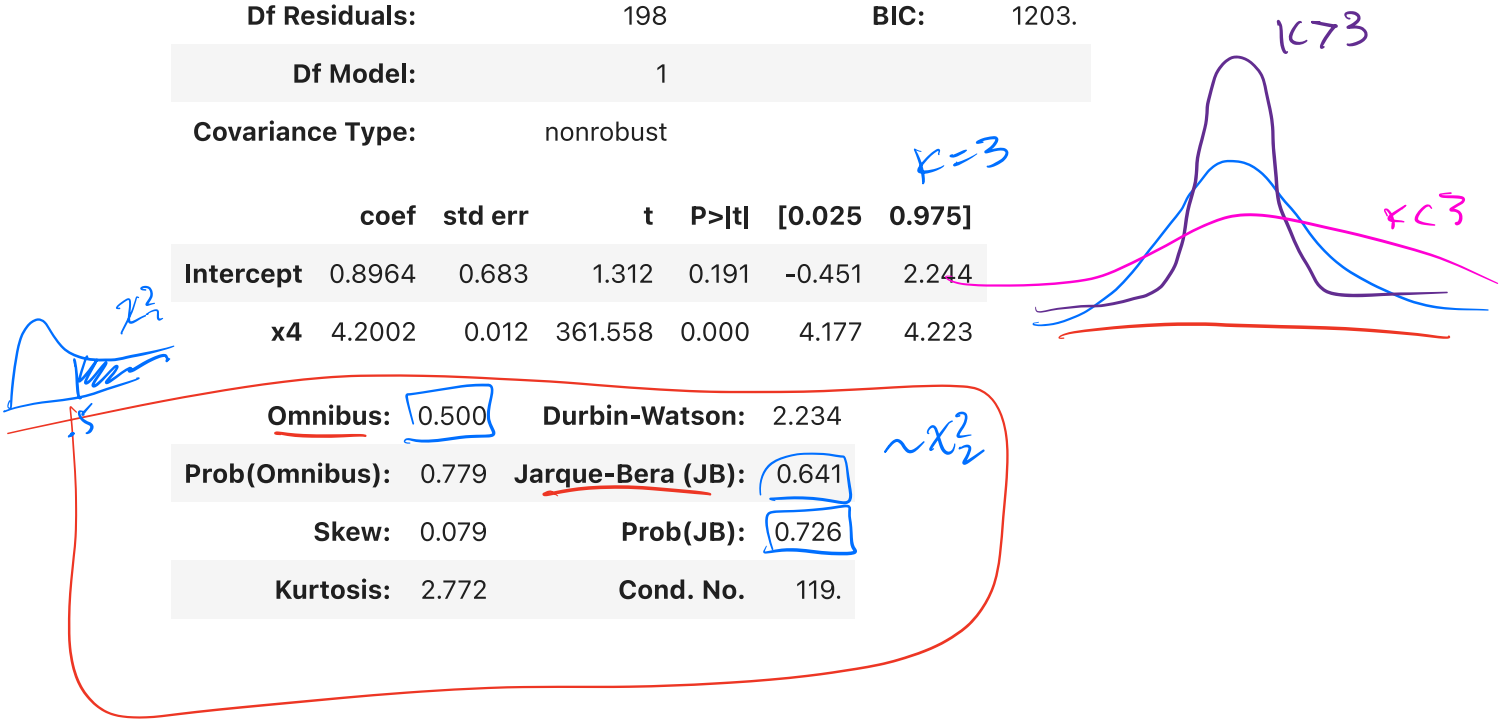
Out[60]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.998
Model:	OLS	**Adj. R-squared:**	0.998
Method:	Least Squares	**F-statistic:**	1.307e+05
Date:	Wed, 25 Sep 2024	**Prob (F-statistic):**	3.46e-281
Time:	22:02:42	**Log-Likelihood:**	-596.26
No. Observations:	200	**AIC:**	1197.
Df Residuals:	198	**BIC:**	1203.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.8964	0.683	1.312	0.191	-0.451	2.244
x4	4.2002	0.012	361.558	0.000	4.177	4.223

Omnibus:	0.500	**Durbin-Watson:**	2.234
Prob(Omnibus):	0.779	**Jarque-Bera (JB):**	0.641
Skew:	0.079	**Prob(JB):**	0.726
Kurtosis:	2.772	**Cond. No.**	119.



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

5.

Geometric Interpretation of the Models

Model 1:

$y \sim X_1 + X_2 + X_3 + res4$:

Here, $res4$ is the vector that represents the part of

X_4 that is orthogonal to $span\{X_1, X_2, X_3\}$. Since it is orthogonal to $span\{X_1, X_2, X_3\}$, it adds information to the model that $span\{X_1, X_2, X_3\}$ cannot explain. Geometrically, you are using four vectors: X_1, X_2, X_3, and $res4$, form a 4-dimensional basis.

Model 2:

$y \sim X_1 + X_2 + X_3 + X_4$:

In this model, you are using X_4 directly. However, when you regress y onto all predictors $X_1,\ X_2,\ X_3\ , X_4$, any part of

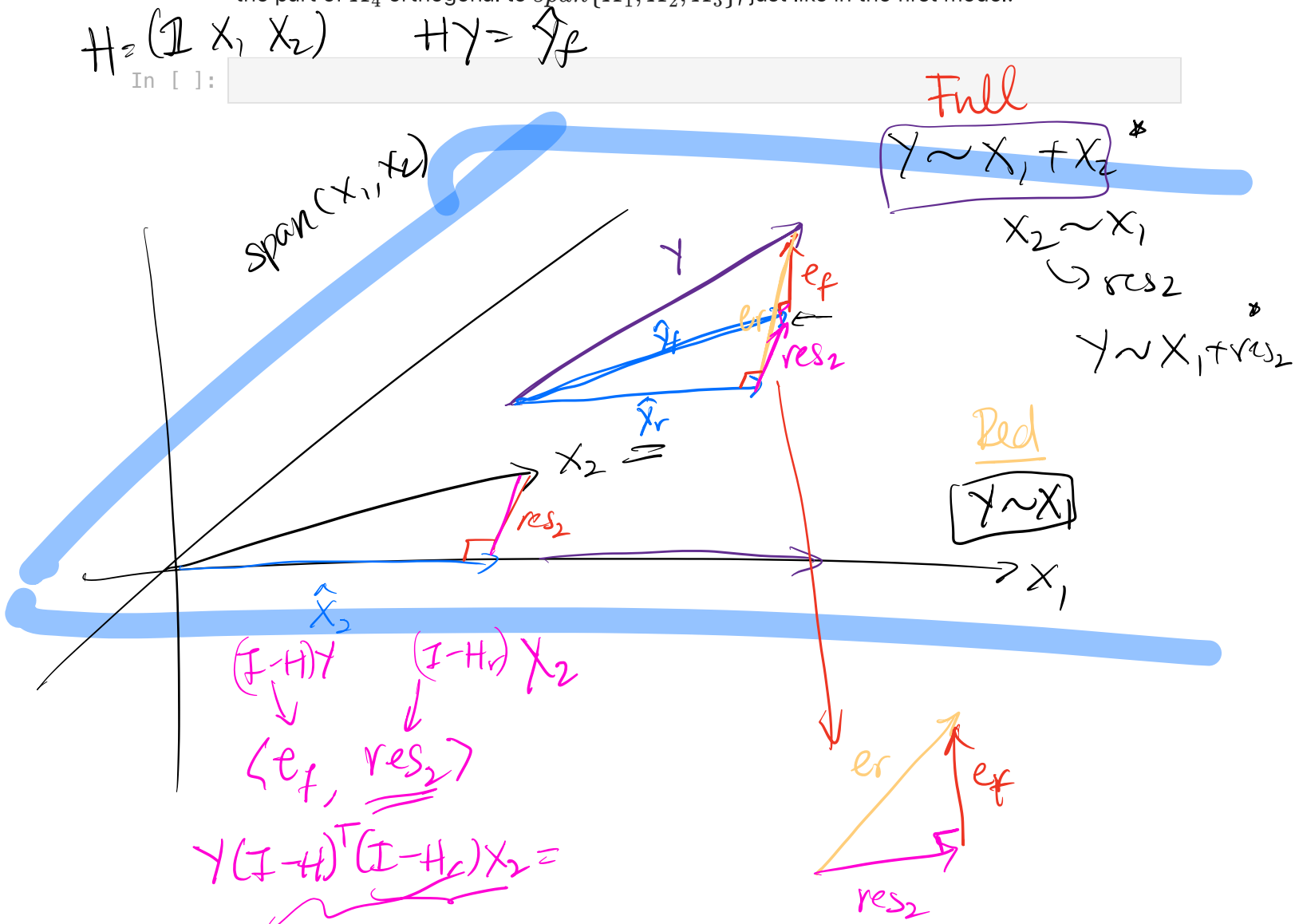
X_4 that is explained by X_1, X_2, or X_3 (the projection of X_2 onto $span\{X_1, X_2, X_3\}$) does not add new information. The model is effectively using the part of X_4 which is orthogonal to $span\{X_1, X_2, X_3\}$ (i.e., the same $res4$ from the first model) to explain y, in addition to X_1, X_2, and X_3.

Why the coefficients are the same:

In both models, the last predictor variable (whether it's $res4$ in Model 1 or X_4 in Model 2) represents the same direction in the vector space: the part of X_4 that is orthogonal to $span\{X_1, X_2, X_3\}$

In $y \sim X_1 + X_2 + X_3 + res4$, you're explicitly using the orthogonal part of X_4 (i.e., $res4$). In $y \sim X_1 + X_2 + X_3 + X_4$, although you include all of X_4, the model implicitly separates the part of X_4 that is redundant (explained by $\{X_1, X_2, X_3\}$) and only uses the part of X_4 orthogonal to $span\{X_1, X_2, X_3\}$, just like in the first model.

In [ ]:



$$\|res_2\|^2 + \|e_f\|^2 = \|e_r\|^2$$

$$\boxed{\|res_2\|^2} + SSE_f = SSE_r$$

$$SS(X_2|X_1) + \|res_2\|^2 = SSE_r - SSE_f$$

$$F = \frac{SS(X_2|X_1)/1}{\hat{\sigma}^2} \sim t^2$$

$$res_2 = X_2 - H_{(X_1)}X_2 = (I - H_{(X_1)})X_2$$

$$e_f = Y - HY = (I-H)Y$$

$$\langle e_f, res_2 \rangle = Y(I-H)(I-H_{(X_1)})X_2$$
$$= Y(I - H - H_{(X_1)} + HH_{(X_1)})X_2$$
$$= Y(I - H - H_{(X_1)} + H_{(X_1)})X_2 = Y(I-H)X_2 = Y(X_2 - HX_2) = Y(X_2 - X_2) = 0$$

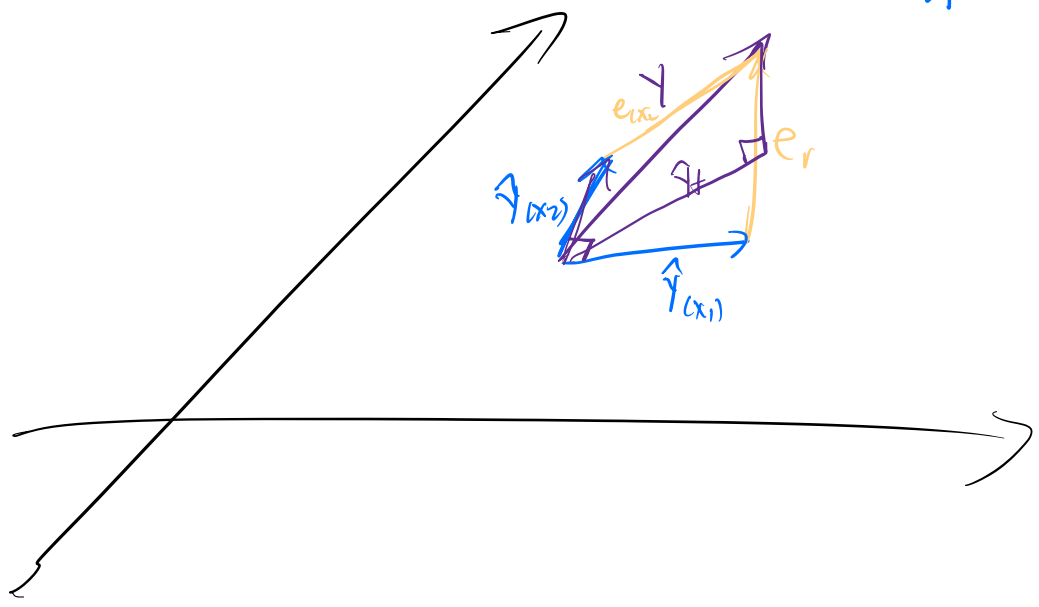
why is $HH_{X_1} = H_{X_1}$?

$HH_{(X_1)} = HX_1(X_1^TX_1)^{-1}X_1^T$

$= X_1(X_1^TX_1)^{-1}X_1^T = H_{(X_1)}$

$X_1 \in span(X_1, X_2) \Rightarrow HX_1 = X_1$

If $X_1 \perp X_2$,



$X_2 \sim X_1 \Rightarrow$

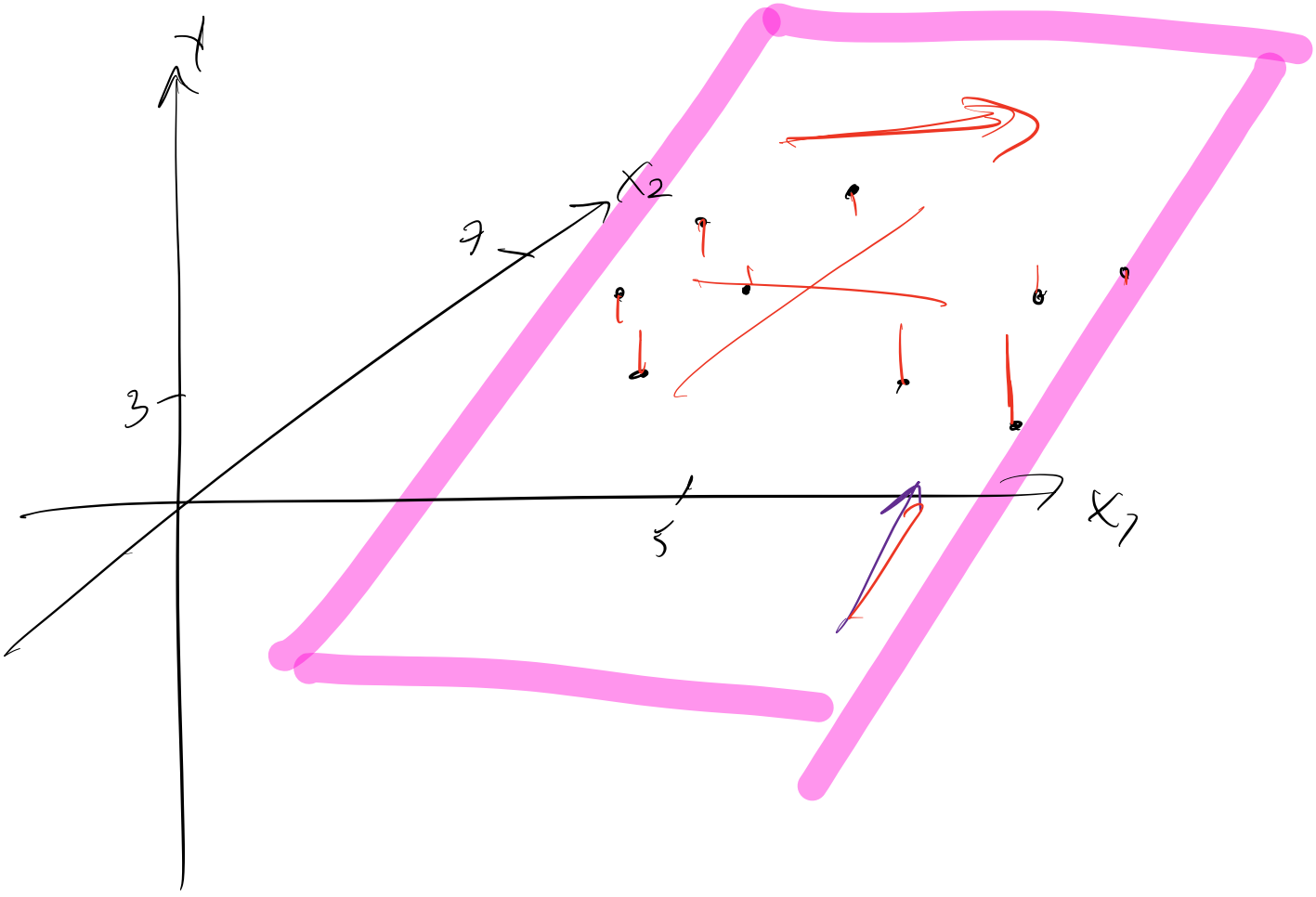
$res_2 = X_2 - \bar{X}_2$

$= X_2$ shifted by a constant

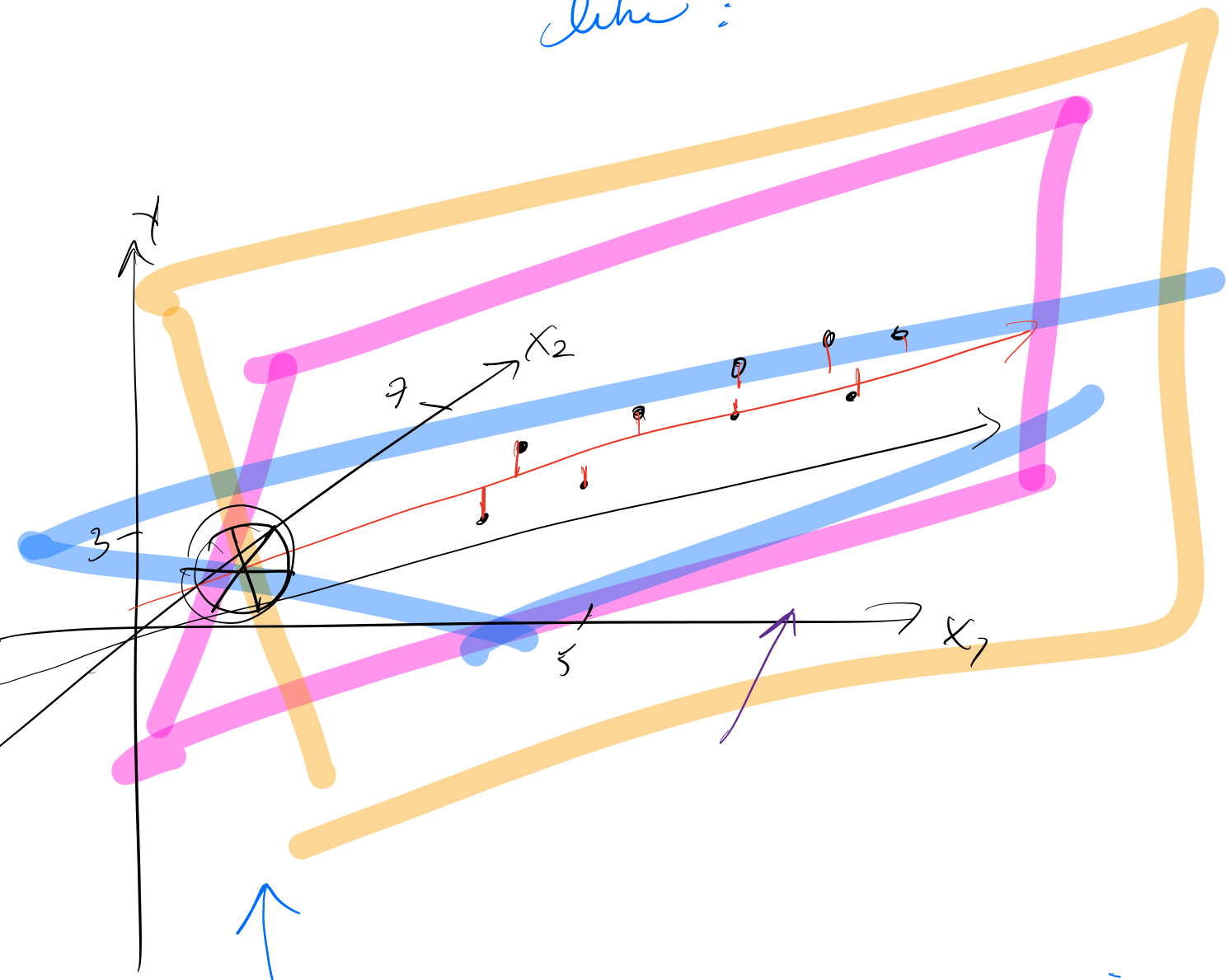
Understanding Multicollinearity with Spaces

If $X_1 \perp X_2$, then

$Y \sim X_1 + X_2$ looks like:



If X_1 & X_2 are very collinear, then $y \sim X_1 + X_2$ looks like:



many planes result in similar SSEs! → small changes in X result in large changes in estimated slopes $\hat{\beta}$!