Linear and Logistic Regression

EBA-3530 Causality, Machine Learning and Forecasting

1 Linear Regression

1.1 Create histograms for income, education, and prestige. Comment the histogram.

Following observations for the variables can be made based on the *Figure 1.1*:

Income and prestige variables have a high correlation. The histogram presents almost the same distribution: both are relatively symmetric; both histograms also reveal the concentration in groups at the low-income/rate-of-prestige and high-income/rate-of-prestige levels, indicating occupations are targeted around the two extreme levels where the middle level is comparatively rare. For education, there are two distinct high bars. It is best to say that there are two groups of occupations, with one requiring a high education and the other relatively low. Education overall shows that many occupations have moderate education levels. Prestige interestingly shows a polynomial-like distribution. This implies that extremely prestigious or non-prestigious occupations are common.

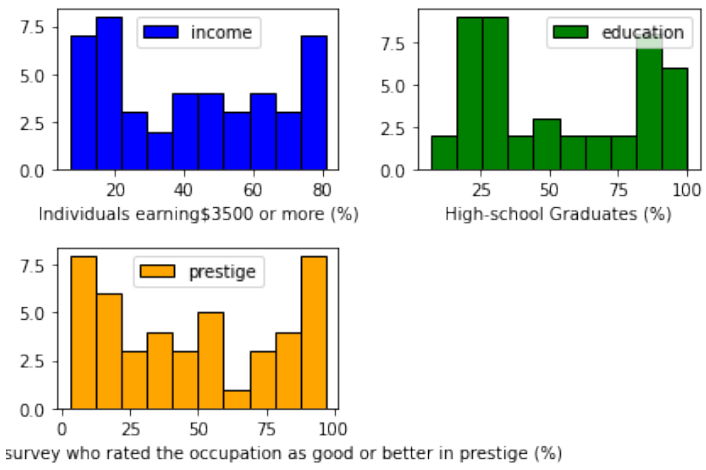


FIGURE 1.1 Histograms for income, education, and prestige from dataset Duncan

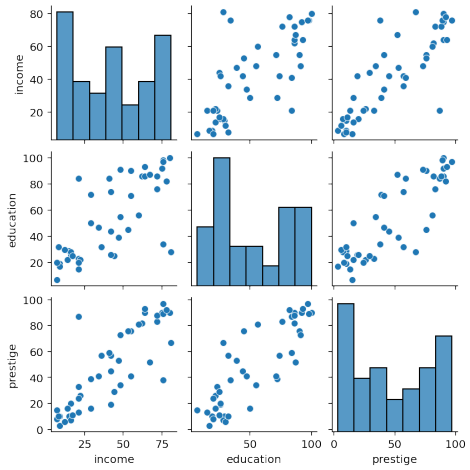


FIGURE 1.2 Scatter plots for income, education, and prestige from dataset Duncan

1.2 Create scatter plots for income vs prestige and education vs prestige. Comment the plots.

Following observations for the variables can be made based on the *Figure 1.2*:

The above-mentioned assumption that "income vs. prestige" is assured In its scatter plot, we can see a positive relationship that follows an upward diagonal direction with very few outliers. In "education vs. income" and "education vs. prestige", we can observe that in both scatter plots, the middle level (around 50) both presents scattered and abnormal distances from each value. The abnormal distribution of data points can be concluded to show weak association despite the fact that it has a rough upward diagonal shape. However, we can conclude that this highlights the fact that income vs. prestige has the strongest relationship out of the three.

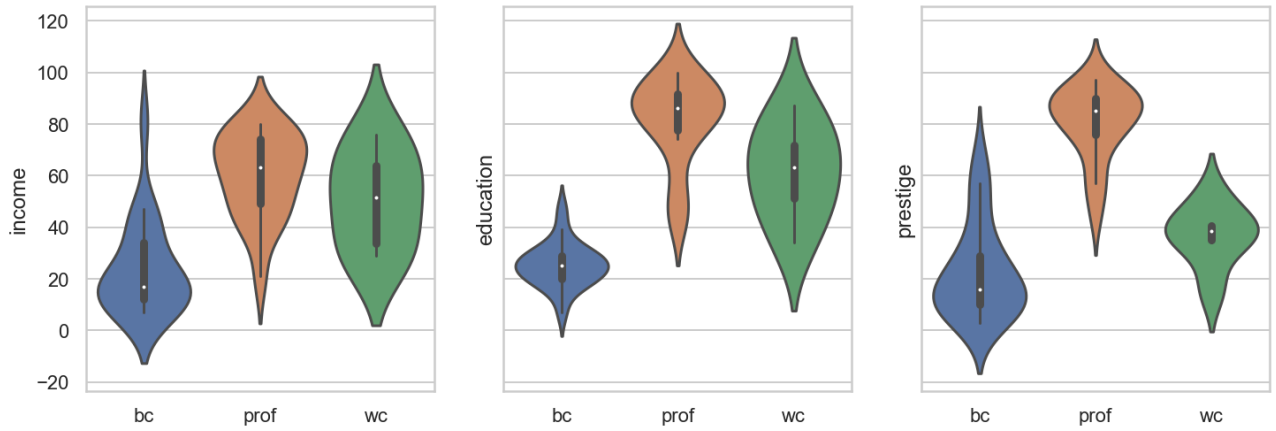


FIGURE 1.3 Violin plots for income, education, and prestige from dataset Duncan

1.3 Create violin plots for income, prestige, and education. Each of these plots, display a violin for each of the 3 categories prof, bc, and wc.

Following observations for the variables can be made based on the *Figure 1.3*:

Here, the results end up being nearly identical to the earlier results. Let us take a look at each profession level in three categories altogether. For "bc" (blue-collar) level occupations, all of its three violin plots have peaked at the bottom level, representing that the "bc' group tends to receive a lower prestige score, have a lower income, and require a lower education level. In "wc" (white-collar), income and education level have a wider range of peaks. "wc" occupations, as predicted, lie in between "bc" and 'pro' occupation prestige levels, which peak around 40. Lastly, we close this section with an interesting thought on the violin plots of "pro" (professional). It can be seen that in income and education, "wc" and 'pro' actually do share similarities, whereas 'pro' plots peak distinctly on the upper levels across all three plots. It shares the common assumption that high-educational occupations tend to have higher income levels and prestige scores.

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-6.0647	4.272	-1.420	0.163	-14.686	2.556
income	0.5987	0.120	5.003	0.000	0.357	0.840
education	0.5458	0.098	5.555	0.000	0.348	0.744

TABLE 1.1 OLS Regression Results from Equation I.1

1.4 Write down the estimates for β_i for $i = 0,1,2$.

We run the OLS Regression from `statsmodels` package and with *Equation I.1*

$$prestige = \beta_0 + \beta_1 income + \beta_2 education + \epsilon \qquad \text{(I.1)}$$

and gain our result as from *Table 1.1*: β_0 = -6.0647, β_1 = 0.5987, β_2 = 0.5458.

1.5 What are the p-values for the 3 coefficient estimates? Are they significant at the 5% level?

> Again, as a result from *Table 1.1*: intercept = 0.163, income = 0.000, education = 0.000. The choice between null and alternative hypotheses hinges on a sample statistic that quantifies the gap between the null truth and the estimated parameters derived from the sample. Here, both our variables, income and education, reject the null hypothesis and are significant at the 5% level. Our intercept, however, is not significant at the 5% level ($p > 0.05$). It is probably due to small amount of data, thus not able to reject the hypothesis that $\beta_0 = 0$.

1.6 What is interpretation of the education coefficient? Be concrete.

> The coefficient of education is 0.5458, where $\beta_0 > 0$. It states that every one additional unit of education will increase 0.5458 units of our Y variable (prestige).

1.7 Calculate the error terms $\epsilon = y_i - \hat{y}_i$ and make a histogram of the error terms. Do errors behave according to OLS theory? You are expected to calculate the errors by yourself, and not using an in-build function.

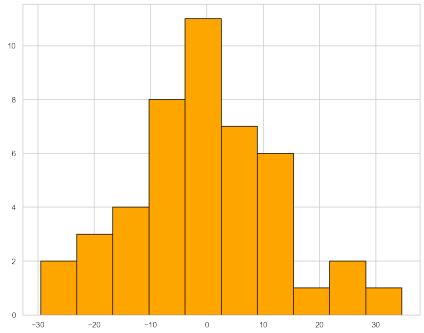


FIGURE 1.4 Histogram for error terms for OLS regression model from Equation 1.1

We can begin with the basics of frequentist uncertainty, the assumptions made about the error terms, and the statistical properties of the OLS estimators. In our model, we made the presumption that the residuals follow that $\epsilon \sim N(o, \sigma^2)$ where $\text{Var}(\epsilon) = \sigma^2$ and $\mu_\epsilon = E(\epsilon) = 0$. As a result of *Figure 1.4*, the ever-important central limit theorem (CLT) states that when the random variables are large enough, the sum will converge to a normal distribution. Here, we do observe the likely shape of a normal distribution, which behaves according to the OLS theory we have assumed above.

2 Logistic Regression

2.1 Creating an extra column in the Duncan's data set where y = 1 if type = *prof*, else y = 0.

	type	income	education	prestige	popf_bin
contractor	prof	53	45	76	1
factory.owner	prof	60	56	81	1
store.manager	prof	42	44	45	1
banker	prof	78	82	92	1
bookkeeper	wc	29	72	39	0
mail.carrier	wc	48	55	34	0
insurance.agent	wc	55	71	41	0
store.clerk	wc	29	50	16	0
carpenter	bc	21	23	33	0
electrician	bc	47	39	53	0
RR.engineer	bc	81	28	67	0
machinist	bc	36	32	57	0

TABLE 2.1 Excerpt result from Duncan's data

As shown in *Table 2.1*, the required result is presented in column "popf_bin" where if type is 'prof' then equal to 1 and 0 to other values in type.

2.2 Fit the following logistic regression:

$$log(\frac{Pr(y=1|x)}{1-Pr(y=1|x)}) = \beta_0 + \beta_1 income + \beta_2 education + \beta_3 prestige \qquad \text{(I.2.1)}$$

Dep. Variable:	popf_bin	**No. Observations:**	45
Model:	GLM	**Df Residuals:**	41
Model Family:	Binomial	**Df Model:**	3
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-4.6175
Date:	Mon, 03 Apr 2023	**Deviance:**	9.2350
Time:	20:45:44	**Pearson chi2:**	16.1
No. Iterations:	8	**Pseudo R-squ. (CS):**	0.6804
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-10.3815	4.789	-2.168	0.030	-19.767	-0.996
income	-0.0988	0.074	-1.332	0.183	-0.244	0.047
education	0.0548	0.042	1.300	0.194	-0.028	0.138
prestige	0.2109	0.092	2.301	0.021	0.031	0.391

Observations are based on the following: we use the logit link function as our depend variable , 'popf_bin' is either 0 or 1, with the required independent variables of income, education, and prestige.

TABLE 2.2 GLM Regression Results from Equation I.2.1

2.3 Write down all coefficient estimates. Are they significant at the 5% level?

We run the GLM Regression with logit link function from `statsmodels` package with *Equation I.2.1* and gain our coefficient estimates as from *Table 2.2*:

β_0 = -10.3815, β_1 = -0.0988, β_2 = 0.0548, β_3 =0.2109.

The accepted risk of a false discovery for each coefficient is at cutoff α = 0.05. Continuing our former explanation of frequentist uncertainty and hypothesis testing, here we are to examine again whether our variables show significance and can reject our null hypothesis in this logistic regression model. We can see that only 'prestige' variable presents a p = 0.021 which is smaller than 0.05 does reject our null hypothesis and is significant. In 'income' and 'education,' since they both show a 0.05 probability of a false positive (p > 0.05) and we have to accept our null hypothesis where $\beta = 0$.

2.4 Fit the follow models and what are the β_1 estimates and their p-values in the 3 different models?

$$log(\frac{Pr(y=1\,|\,x)}{1-Pr(y=1\,|\,x)}) = \beta_0 + \beta_1 income \quad \text{(a)}$$

$$log(\frac{Pr(y=1\,|\,x)}{1-Pr(y=1\,|\,x)}) = \beta_0 + \beta_1 education \quad \text{(b)}$$

$$log(\frac{Pr(y=1\,|\,x)}{1-Pr(y=1\,|\,x)}) = \beta_0 + \beta_1 prestige \quad \text{(c)}$$

Dep. Variable:	popf_bin	**No. Observations:**	45
Model:	GLM	**Df Residuals:**	43
Model Family:	Binomial	**Df Model:**	1
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-20.641
Date:	Mon, 03 Apr 2023	**Deviance:**	41.282
Time:	23:14:00	**Pearson chi2:**	42.5
No. Iterations:	5	**Pseudo R-squ. (CS):**	0.3486
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-3.5220	1.018	-3.460	0.001	-5.517	-1.527
income	0.0689	0.020	3.503	0.000	0.030	0.107

TABLE 2.3 GLM Regression Results from Equation (a)

Dep. Variable:	popf_bin	**No. Observations:**	45
Model:	GLM	**Df Residuals:**	43
Model Family:	Binomial	**Df Model:**	1
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-12.767
Date:	Mon, 03 Apr 2023	**Deviance:**	25.534
Time:	23:15:12	**Pearson chi2:**	29.6
No. Iterations:	6	**Pseudo R-squ. (CS):**	0.5409
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-5.6491	1.515	-3.730	0.000	-8.617	-2.681
education	0.0906	0.023	3.923	0.000	0.045	0.136

TABLE 2.4 GLM Regression Results from Equation (b)

Dep. Variable:	popf_bin	**No. Observations:**	45
Model:	GLM	**Df Residuals:**	43
Model Family:	Binomial	**Df Model:**	1
Link Function:	Logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-7.1377
Date:	Mon, 03 Apr 2023	**Deviance:**	14.275
Time:	23:16:04	**Pearson chi2:**	15.4
No. Iterations:	7	**Pseudo R-squ. (CS):**	0.6426
Covariance Type:	nonrobust		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-8.7383	3.033	-2.881	0.004	-14.684	-2.793
prestige	0.1510	0.052	2.911	0.004	0.049	0.253

TABLE 2.5 GLM Regression Results from Equation (c)

We run the GLM Regression with logit link function from `statsmodels` package with *Equation (a)*, *(b)*, and *(c)* and gain our coefficient estimates and p-values as from *Table 2.3* to *2.5*:

$\beta_1 income = 0.0689$, $\beta_1 education = 0.0906$, $\beta_1 prestige = 0.1510$.

$p - values_{(a)} = 0.000$, $p - values_{(b)} = 0.000$, $p - values_{(c)} = 0.004$.

2.5 Which of the 4 models makes the most sense for classifying whether a person has a managerial type job? Argue (briefly) from a statistical point of view.

model	AIC	Pseudo R^2
Equation$_{I1}$	17.235	0.6804
Equation$_a$	45.282	0.3486
Equation$_b$	29.534	0.5409
Equation$_c$	18.275	0.6426

TABLE 2.6 Model Comparison Results

Table 2.6 lists model selection criteria for all models. In selection, we can first remove Equation (a) and Equation (b) models where they show high AIC scores and low Pseudo R^2 scores. Here we have a constraint: classifying whether a person has a managerial type job. First, by the p-values in *Table 2.2*, we see that only 'prestige' rejects the null hypothesis and shows significance (referring to question 2.3). They imply that they might not have a meaningful or substantial impact on the dependent variable in our model. However, as the AIC score and Pseudo R^2 score show, this model can end up with the significant variable, which is "prestige," which would be our Equation (c) model. This model shows significant variability in its variables and has the second lowest AIC score with a high Pseudo R^2 score. Moreover, variable 'prestige' in accordance with the plots do agree that higher prestige levels do share higher education levels and income. Meaning the prediction for a managerial position could be of use.

Bootstrapping and Autoregressive Models

EBA-3530 Causality, Machine Learning and Forecasting

3 Uncertainty in Linear Models

3.1 For $N = 20$, generate x_1, x_2 and x_3 as suggested in the appendix.

As the appendix suggested, we generate our variables x1, x2, x3 with,

$$x \sim \mathcal{N}(\mu, \Sigma), \text{ where } \mu = \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & 0.4 & 0 \\ 0.4 & 1.5 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

```
x1: [4.36374315, 4.78005043, 4.94450806, 5.84325979, 5.88522661,
       5.92188098, 4.57082527, 5.06325986, 3.5198066 , 4.11901961,
       5.22283911, 5.57565248, 2.92542633, 5.45094481, 4.85000315,
       5.75498172, 3.60739389, 2.42002217, 5.45026833, 5.74967708]

x2: [2.20768854, 1.95944174, 2.31057042, 3.76992373, 1.6214196 ,
       2.9105985 , 1.81443434, 3.4012444 , 0.22396532, 2.97843056,
       3.17868247, 1.4708037 , 1.54639316, 1.06150187, 2.15154362,
       1.16128924, 1.34093718, 0.17975934, 1.28869353, 1.81090254]

x3: [ 4.7923386 , 0.19442543, 0.68006288, 0.20265144, 2.43798989,
      1.62120249, 0.91044842, -1.03317302, 1.29333433,  0.89976909,
      2.84084466, 1.02155662, 0.96832803, -1.42269635,  1.94078026,
      2.58317303, 0.05871876, 0.05491707, -0.3032869 , -0.97248063]
```

3.2 Generate the true y values using Equation I.3.1.

$$y = 10 + x_1 + 5x_2 + 3x_3 + \epsilon \qquad \text{(I.3.1)}$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon \qquad \text{(I.3.2)}$$

We generate the true y values using *Equation I.3.1* with Question (1.1) generated x1, x2, and x3 independent values.

```
y: [
21.37079885 20.70187859 21.77171235 21.70828109 20.88367959
20.77186839 22.84779021 21.57692015 21.27575571 22.00768464
21.81867369 23.23260999 21.98384252 22.39241745 23.13040914
21.16095041 21.1294466  20.71343248 20.90381482 21.55640526
]
```

3.3 Generate B = 1000 bootstrap data sets $\{x, y\}_{b=1}^{B}$ using sampling with replacement for $N = 20$.

Here we have generated bootstrap data sets $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \ldots, \hat{\beta}_{1000}$ where $\{\hat{\beta}_1\}_{b=1}^{B}$ is an approximation to the sampling distribution for $\hat{\beta}$. Due to the large amount of data sets, we have list one set of bootstrap data set

from `bootstrap_data[999]` we get $\hat{\beta}_{1000}$:

```
array([4.36374315, 5.45094481, 2.42002217, 2.42002217, 4.57082527,
   5.57565248, 5.92188098, 4.85000315, 2.42002217, 2.42002217,
   5.84325979, 3.60739389, 4.57082527, 5.74967708, 2.42002217,
   5.92188098, 2.42002217, 5.22283911, 4.36374315, 4.78005043]),

array([2.20768854, 1.06150187, 0.17975934, 0.17975934, 1.81443434,
   1.4708037 , 2.9105985 , 2.15154362, 0.17975934, 0.17975934,
   3.76992373, 1.34093718, 1.81443434, 1.81090254, 0.17975934,
   2.9105985 , 0.17975934, 3.17868247, 2.20768854, 1.95944174]),

array([
21.37079885, 22.39241745, 20.71343248, 20.71343248, 22.84779021,
23.23260999, 20.77186839, 23.13040914, 20.71343248, 20.71343248,
21.70828109, 21.1294466 , 22.84779021, 21.55640526, 20.71343248,
20.77186839, 20.71343248, 21.81867369, 21.37079885, 20.70187859])]
```

The first array is a resample of the x1 variable; the second from the x2 variable; and the third is from our true y variable. Note that here we did not resample the x3 variable into our bootstrap data sets since the following questions using our data sets do not require the x3 variable. For more, *see the appendix*.

3.4 For each bootstrap sample b, estimate all $\beta_{b=1}^{B}$ parameters in Equation I.3.2.

Here we are asked to estimate each sample $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \ldots, \hat{\beta}_{1000}$ for their parameters. We first create `boot_fn` function to stack our independent variables then fit them with the according sample b set y for linear regression and return intercept (β_0 in I.3.2) and coefficients (β_1 and β_2 in I.3.2). For each set, their estimate parameter are appended in the list named `coef_list`. Here, we list the first 10 estimates which is the model for data set: $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \ldots \hat{\beta}_5$. For more, *see the appendix*.

	Intercept β_0	**Coefficient β_1**	**Coefficient β_2**
$\hat{\beta}_1$	19.2449632639944	0.57917424	-0.00199055
$\hat{\beta}_2$	20.7726583111640	0.02537041	0.34661821
$\hat{\beta}_3$	21.0792310240891	0.12784398	-0.0964879
$\hat{\beta}_4$	19.6174592515887	0.54312768	-0.24276874
$\hat{\beta}_5$	22.3244821597371	-0.17317979	0.10204514
$\hat{\beta}_6$	21.2384568157719	0.15542026	-0.0690385
$\hat{\beta}_7$	20.4032705814967	0.24976247	0.1589486
$\hat{\beta}_8$	21.0050664366594	0.08814184	0.00626534
$\hat{\beta}_9$	21.1598844203662	-0.15202802	0.37870109
$\hat{\beta}_{10}$	20.6571381405263	0.13479513	0.02202783

TABLE 3.1 Estimates from Each Bootstrap Sample b

3.5 Plot histograms for all bootstrap parameters β_i and comment briefly on the results.

Following observations for the parameters estimates can be made based on the *Figure 3.1*:

Before taking a look at the estimates histograms, *Table 3.2* lists the calculated "mean" $\bar{\beta}_i$ and "standard errors" $se(\hat{\beta}_i)$. Now, looking at the histograms, we can observe the close accuracy to our calculated means, which is likely where the peak is. The histograms presents general normal distributed resemblance. Here we can take a guess; the Central Limit Theorem (CLT) suggests these sampling distributions are likely to be approximately normally distributed.

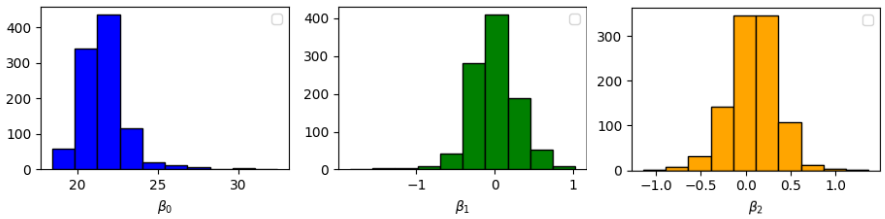


FIGURE 3.1 Histograms for β_0, β_1, and β_2 Estimates from bootstrap dataset of Equation I.3.2

	Mean	Standard Error $se(\hat{\beta}_i)$
β_0	21.62046040617432	1.4687315852743303
β_1	-0.015247733550798	0.3070997593678382
β_2	0.0786790585427962	0.2700938434323735

TABLE 3.2 β_0, β_1, and β_2 Estimates of Mean and Standard Deviation from bootstrap datasets of Equation I.3.2

3.6 Repeat exercise 3.1 and 3.2 but for N = 1000.

As the appendix suggested, we generate our variables x1, x2, x3 with *Equation I.3.1*,

$$\mathrm{x} \sim \mathcal{N}(\mu, \Sigma), \text{where } \mu = \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & 0.4 & 0 \\ 0.4 & 1.5 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

However, with sample size of 1000 instead of 20.

```
x1: array([4.36374315, 4.78005043, 4.94450806, 5.84325979, 5.88522661,
                                      ⋮
           5.10237728, 6.83374034, 5.54639803, 5.09140696, 4.40942887])

x2: array([
      2.20768854e+00,  1.95944174e+00,  2.31057042e+00, 3.76992373e+00,
                                      ⋮
      2.18858813e+00,  2.70140563e+00,  2.68221961e+00,  1.67459073e+00])

x3: array([
      4.79233860e+00,  1.94425432e-01,  6.80062884e-01,  2.02651444e-01,
                                      ⋮
      8.15745051e-01,  7.79194345e-01,  3.77541035e-01,  1.42802397e-02])
```

Due to the large amount of sample sizes, for the complete answers *see appendix*.

We generate the true y values using *Equation I.3.1* with Question (1.6) generated x1, x2, and x3 independent values. For more, *see appendix*.

```
y: [
21.03782171, 23.12332523, 24.22451326, 22.1943038 , 23.86686251,
                                  :
23.49370929, 22.50481386, 24.51534409, 23.36786655, 22.58662623
]
```

3.7 Then repeat exercise 3.3-3.5. Comment briefly on the difference between the histograms you obtain now and the histograms you obtained before.

Here we have generated bootstrap data sets $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \ldots, \hat{\beta}_{1000}$ where $\{\hat{\beta}_1\}_{b=1}^{B}$ is an approximation to the sampling distribution for $\hat{\beta}$. Due to the large amount of data sets, we have list one set of bootstrap data set

from `bootstrap_data[999]` we get $\hat{\beta}_{1000}$:

```
array([
4.55436068, 6.27383617, 3.31955654, 6.27383617, 3.7037587 ,
3.61293893, 5.0814203 , 3.31955654, 3.32309186, 3.61293893,
4.91926121, 6.27383617, 4.4750466 , 3.32309186, 3.31955654,
3.61293893, 6.27383617, 4.53185395, 5.0814203 , 3.31955654]),

array([
8.96917843, 22.09275025, 16.1238491 , 22.09275025, 10.08982448,
8.16337428, 18.10309984, 16.1238491 , -4.92615148,  8.16337428,
5.04042304, 22.09275025,  4.91704707, -4.92615148, 16.1238491 ,
8.16337428, 22.09275025, 23.07055015, 18.10309984, 16.1238491 ]),

array([
24.77880398, 23.66173979, 22.4246014 , 23.66173979, 22.89817221,
22.95138413, 23.05254472, 22.4246014 , 23.39069908, 22.95138413,
21.5562368 , 23.66173979, 22.97616066, 23.39069908, 22.4246014 ,
22.95138413, 23.66173979, 22.15799115, 23.05254472, 22.4246014 ])
```

The structure of our sample list follows the same structure as Question 1.3 However, now our sample size for resampling is 1000 instead of 20 to withdraw from. For more, *see the appendix*.

For each set, their estimate parameters are appended in the list named `coef_list2`. Here, we are showing the above `bootstrap_data[999]` which is the 1000th model resampling set ($\hat{\beta}_{1000}$) as example.

from `coef_list2[999]` we get the model for this sample set's $\beta_0, \beta_1, \beta_2$:

	Intercept	Coefficient β_1	Coefficient β_2
$\hat{\beta}_{1000}$	21.6908746090675	0.39022535	-0.03043119

TABLE 3.3 β_0, β_1, and β_2 Estimates of Parameters from $\hat{\beta}_{1000}$ bootstrap dataset of Equation I.3.2

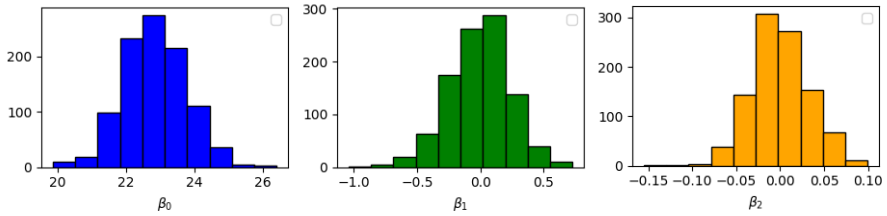


FIGURE 3.2 Histograms for β_0, β_1, and β_2 Estimates from bootstrap dataset of Equation I.3.2

	Mean	Standard Error $se(\hat{\beta}_i)$
β_0	22.831278765656428	0.9225771909971409
β_1	-0.0074717953423585805	0.2422684405468664
β_2	0.0004974017554233064	0.03230157943553694

TABLE 3.4 β_0, β_1, and β_2 Estimates of Mean and Standard Deviation from bootstrap datasets of Equation I.3.2

After repeating the exercise 3.3 to 3.5 using N = 1000 instead of N = 20, here are the following observation:

The *Figure 3.2* histogram has larger resamples. This introduces variability into our bootstrap data sets. The with-replacement sampling yields much smoother, accurate results in histograms in conjunction with our Mean Value and Standard Errors shown in *Table 3.4*. The shape of normally distributed bins is much clearer and has narrower/taller bins. The higher level of precision can be observed in the standard error values. In *Table 3.2*, β_0, β_1, and β_2 have the standard errors of roughly 1.5, 0.3, 0.3 respectively. In *Table 3.4*, β_0, β_1, and β_2 have standard errors of roughly 0.9, 0.2, 0.03 respectively. The significantly smaller standard error leads to the conclusion of a reduction in the spread in distribution.

3.8 Use statsmodels and fit the linear regression model in equation 3. Comment on the results.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 (\frac{x_1 + x_2}{2}) + \epsilon \qquad \text{(I.3.3)}$$

$$log(y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 (x_1 x_2) + \epsilon \qquad \text{(I.3.4)}$$

Dep. Variable:		y		**R-squared:**	0.021	
Model:		OLS		**Adj. R-squared:**	-0.094	
Method:		Least Squares		**F-statistic:**	0.1849	
Date:		Mon, 10 Apr 2023		**Prob (F-statistic):**	0.833	
Time:		00:07:01		**Log-Likelihood:**	-22.544	
No. Observations:		20		**AIC:**	51.09	
Df Residuals:		17		**BIC:**	54.07	
Df Model:		2				
Covariance Type:		nonrobust				
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	21.3257	0.889	23.999	0.000	19.451	23.201
x1	0.0053	0.187	0.028	0.978	-0.389	0.400
x2	0.0797	0.195	0.409	0.688	-0.332	0.491
x3	0.0425	0.073	0.582	0.568	-0.112	0.197

TABLE 3.5 OLS Regression Results from Equation (I.3.3)

Dep. Variable:		log(y)		**R-squared:**	0.143	
Model:		OLS		**Adj. R-squared:**	-0.017	
Method:		Least Squares		**F-statistic:**	0.8932	
Date:		Mon, 10 Apr 2023		**Prob (F-statistic):**	0.466	
Time:		00:38:40		**Log-Likelihood:**	40.507	
No. Observations:		20		**AIC:**	-73.01	
Df Residuals:		16		**BIC:**	-69.03	
Df Model:		3				
Covariance Type:		nonrobust				
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	2.9880	0.062	48.342	0.000	2.857	3.119
x1	0.0169	0.014	1.225	0.238	-0.012	0.046
x2	0.0653	0.041	1.580	0.134	-0.022	0.153
x3	-0.0125	0.008	-1.501	0.153	-0.030	0.005

TABLE 3.6 OLS Regression Results from Equation (I.3.4)

The model summary is presented in *Table 3.5*. First thing to unpack here is our coefficients: y increase about about 0.00053, 0.0797, 0.0425 for every 1 unit of $x1$, $x2$, and $x3$ respectively. Our independent variables here does not show high modulation and correlation to our dependent variable y. We will further discuss the issue with *Equation I.3.3* in *Question 3.9*. Here, we will follow up with the examination of test statistics and p-values.

The test statistics $z\beta = \frac{\hat{\beta}}{se(\hat{\beta})}$ tells us the relative position of the observation or estimate within the distribution in the standardised way. Variables *x2*, and *x3* presents high z-scores which indicates the estimates are comparatively far from the mean. The p-values more over suggest that all *x1, x2*, and *x3* have to accept null hypothesis where $\beta = 0$. The independent variables are not highly correlated to y. As R^2 highlights, this model can only explains 2.1% of the variation in the dependent variable. Overall, this model should be further altered and does not perform well.

3.9 What is the problem with the model in equation 3 and how can you solve the problem?

For *Equation I.3.3* compared to *Equation I.3.4*, we added the inference variable $x_3 = \frac{x_1 + x_2}{2}$. This model summary is presented in *Table 3.6*. The problem here is the that our inference variable introduces multicollinearity. Based on Chu et al. (2012), when the independent variables are dependent with each other. We can illustrate as:

$$R^2_{y.12} \neq r^2_{y.1} + r^2_{y2} \qquad \text{(I.3.5)}$$

When two variables have high dependency, the explanation capacity of *x1 or x2* could be covered by the other and in result, presents a fully alternated and inaccurate conclusion. Here we introduce the VIF (Variance Inflation Factor) method to examine our model.

$$VIF = \frac{1}{Tolerance} = \frac{1}{1 - R_i^2} \qquad \text{(I.3.6)}$$

	VIF Score	Coefficient
0	24.059295	const
1	inf	x1
2	inf	x2
3	inf	x3

TABLE 3.7 VIF Scores from Model I.3.3

	VIF Score	Coefficient
0	59.967507	const
1	3.004833	x1
2	24.130157	x2
3	31.285471	x3

TABLE 3.8 VIF Scores from Model I.3.4

For Model of *Equation I.3.3*, the VIF score is shown in *Table 3.7*. We can see that our parameters show 'infinite' for our VIF score. This is caused by the perfect correlation between the coefficients. The x3 variable is simply the mean of each of x1 and x2. This causes perfect multicollinearity.

To solve this issue, I have altered the inference variable to $x_3 = x_1 x_2$. Another alteration made is our dependent variable into $log(y)$. The reason for this is that "*For variables that change multiplicatively with other factors, this is usually the log scale*," Matt (2019, Ch. 2). Based on *Table 3.6*, we see tremendous improvement in our regression model. The overall 14.3% is explained by the model and each variables significantly improves. Though the p-values shows they are not significant for rejecting our null hypothesis. The model replaces the intensely high multicollinearity by replacing the inference variable. To further examine here, *Table 3.8* shows the VIF scores for the altered model. There is a clear demonstration of significant lower VIF score from high to low and x3 remains the highest in VIF score as its inference nature. Thus the second model is preferably more accurate and can yield a better result than the first model.

4 Time series and Model Selection

4.1 Plotting the annual flow of the river.

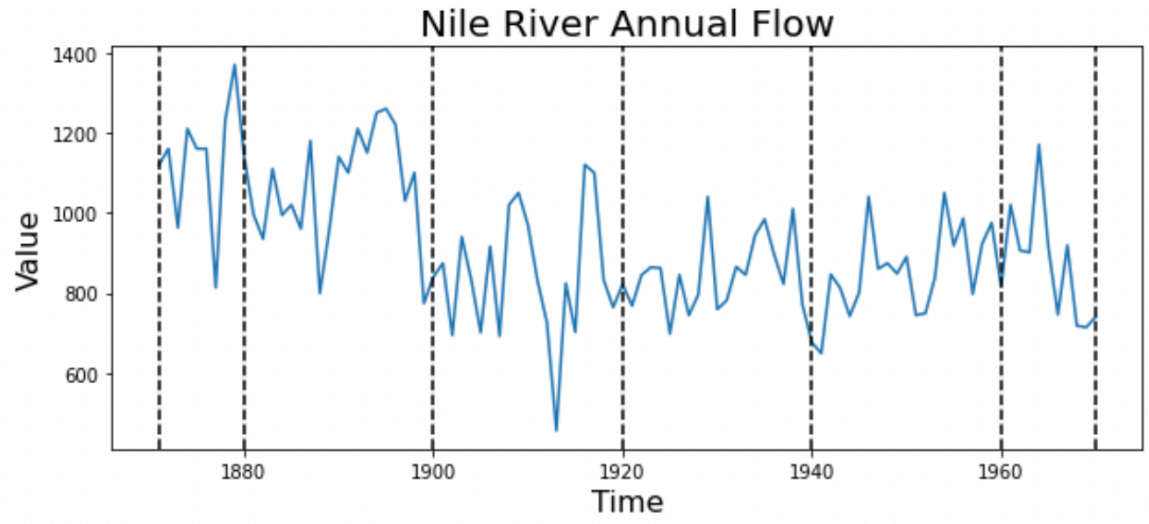


FIGURE 4.1 Nile River Annual Flow

4.2 Then, plot the autocorrelation function (ACF) and partial autocorrelation function (PACF) for 30 lags.

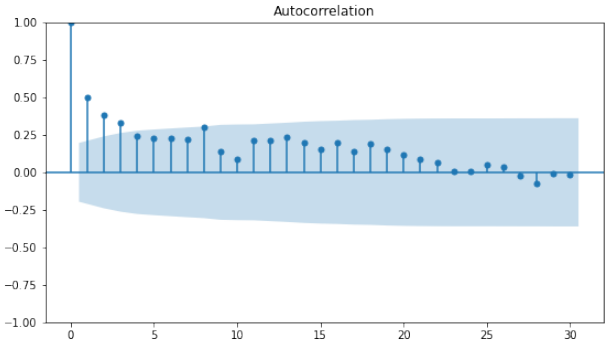


FIGURE 4.2 Autocorrelation Plot with Lags of 30

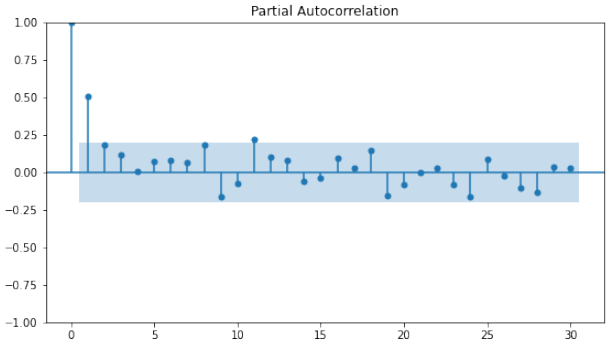


FIGURE 4.3 Partial Autocorrelation Plot with Lags of 30

4.3 What can you say about the annual flow by looking at diagram from exercise 4.1 and by looking to the ACF and PACF from exercise 4.2? Be brief and concrete.

> We can see that our parameters show ‘infinite’ for our VIF score. This is caused by the perfect correlation between the coefficients. The x3 variable is simply the mean of each of x1 and x2. This causes perfect multicollinearity.
>
> From *Figure 4.1*, the annual flow seems to go against the previous year. One year the flow increases and the next it decreases, forming the ups and downs. However, the extent and amount are uncertain and not fixed.

Follow up: starting in *Figure 4.2* we have the following from the autocorrelation function plot: it shows clear decaying movement. Based on the diminishing behaviour, we are dealing with an autoregressive process.

PACF is a main guideline here for determining if the time series can be modelled for AR model. Partial autocorrelation function plot (*Figure 4.3*) suggests we start our autoregressive models with lags 1 and 11, as they show significance.

4.4 Looking to the diagrams from 4.2 select the number of p lags to be used in your AR(p) model. Justify your choice.

The number of p lags selected is based on the PACF plot shown in Figure 4.3. We can observe significance in lags 1 and 11, of which are above the threshold of the confidence interval in the blue area.

We have two models:

$$AR_{(1)} : \hat{y}_1 = \phi_0 + \phi_1 y_{t-1} + \epsilon$$

$$AR_{(11)} : \hat{y}_{11} = \phi_0 + \sum_{i=1}^{11} \phi_i y_{t-i} + \epsilon_{11}$$

Based on textbook MT(ch.2, p.41-57; 61-67), "if $|\beta_1| < 1$, the values are mean reverting." All three models presents $|\beta_1| < 1$. Thus, the method of choosing a model here is via BIC and AIC score. $AR_{(11)}$model presents the lowest score on both and thus are preferred for our forecasting since it is able to fit the data best and strike a balance between the fit and the number of parameters that were used.

4.5 Train an $AR_{(p)}$ model using the p value you chose in question 4.4. After you train the model, forecast the out-of-sample data set and report the root mean squared error (RMSE) for the predictions.

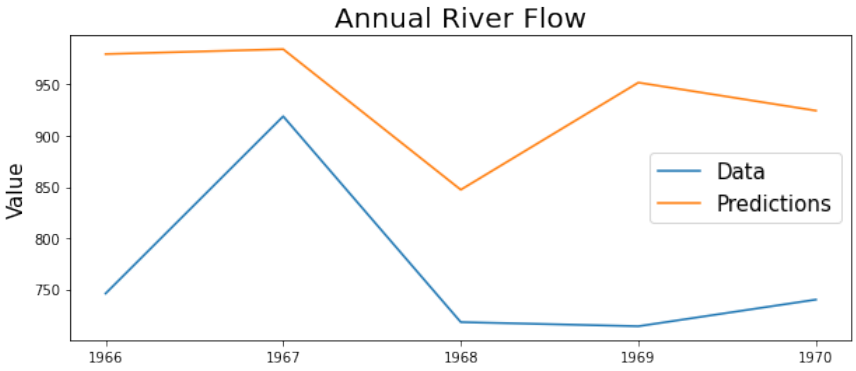


FIGURE 4.4 Nile River Annual Flow Data and Predictions based on AR(11) Model In Comparison

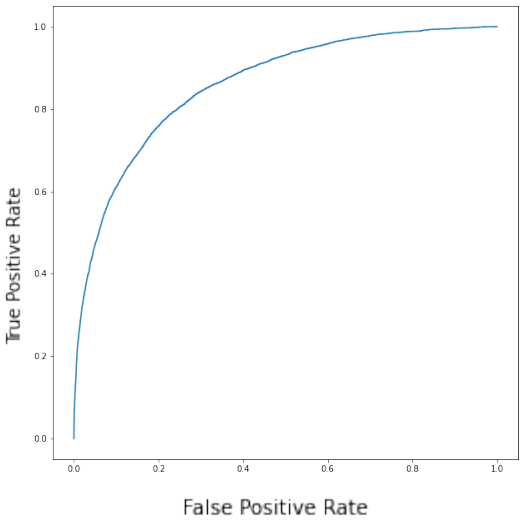
```
Forecasted: [979.776155, 984.534082, 847.285869, 951.937259,
924.530801]

Root Mean Squared Error: 182.38782760372086
```

5 Classification, K-fold Cross-Validation and PCA

5.1 Fitting a logistic regression model using 70%-30% of the data for training-testing the model. Report the area under the roc-curve, simply called AUC, for the test sample.

$$log_{target}(\frac{Pr(y=1|x)}{1-Pr(y=1|x)}) = \beta_0 + \beta_1 var_0 + \beta_2 var_1 + \ldots + \beta_{200} var_{199} \quad \text{(I.5.1)}$$



AUC score in this logistic regression model is `0.8612723617361713` as we can see on the ROC curve plot shown in *Figure 5.1*. The logistic regression is based on *Equation I.5.1* with the provided dataset.csv. The CustomerID in the dataset is dropped by using:

```
df_var = df.filter(like = 'var')
```

FIGURE 5.1 ROC curve for test sample in Logit model I.5.1

5.2 Run a 5-fold cross-validation and report both average and standard deviations for the AUC. Make a nice table summarising your results.

$$CV_n = \frac{1}{n}\sum_{i=1}^{n} Err_i \quad \text{(I.5.2)}$$

We compute the 5-fold cross-validation based on the the result in n number the misclassified observations. Below table presents the averaging values of $Err_1, Err_2, \ldots, Err_n$, where n = 1 to 5. It is important to signify that since we are using LogisticRegression instead of sm.Logit, the additional `penalty = 'none'` is added in order to remove the regularisation by default as required by the exam paper.

$CV_{(1)}$	$CV_{(2)}$	$CV_{(3)}$	$CV_{(4)}$	$CV_{(5)}$
0.84413687	0.84568344	0.84510823	0.85156134	0.85197494

	Score
Average	0.8476929656153205
Standard Deviations	0.0033664355702841

Table 5.1 5-fold CV with AUC Mean and Standard Deviations Based on Logit Model I.5.1

5.3 You use PCA and reduce the original dimensionality to 20, 50, and 100 factors. Apply the PCA transformation to the whole data set.

PCA is applied to reduce dimension of an n x p data matrix X where we reduce to set p = 20 (Right), 50 (Middle), and 100 (Left) in this question.

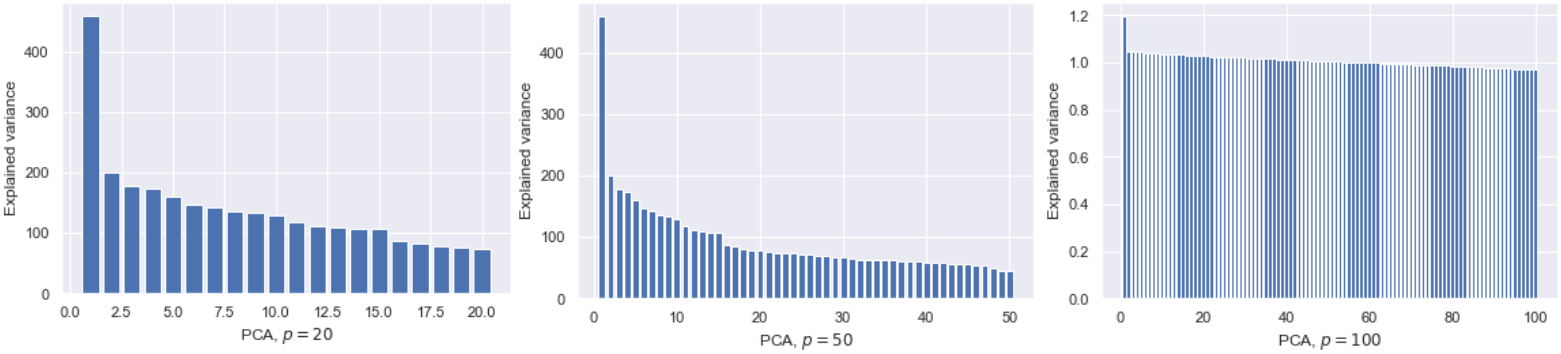


FIGURE 5.2 Explained Variance in Three PCA where p = 20, 50 , and 100

5.4 For each of the data sets in 5.3, conduct a 5-fold cross-validation and fill out *Table 5.2*, where Orig. dim. stands for original dimensionality and are the results from *Exercise 5.2*.

	$Orig.Dim.$	$PCA = 20$	$PCA = 50$	$PCA = 100$
$Avg.AUC$	0.84769296561532	0.60392251116900	0.70503050339282	0.77570295042587
$Std.AUC$	0.00336643557028	0.00372451855854	0.00382611836479	0.00282990109506

Table 5.2 5-fold CV Results based on Exercise 5.3

5.5 Explaining the differences in the approaches used in exercises 5.1, 5.2, and 5.4. Indicate which of the 5 models you have trained the company should use and why.

Approaches differ here because of the advantages of model interpretability and prediction accuracy. The approach of 5-fold cross-validation is to directly gauge test error instead of indirectly, offering advantages over AIC, BIC, adjusted R^2, and C_p by providing assuming less about the true model. PCA is further introduced in *Exercise 5.4* for dimensionality reduction before conducting cross-validation. The PCAs offer an uncomplicated approach for conducting regression with $M < p$ predictors (where M refers to constructing the first M principal components), but they do not function as a method for selecting features; due to that, a linear combination of all p original features is in each of the M principal components of the PCAs. We thus ought to choose the model with the highest AUC score and results in the model of $Orig.Dim.$

6 Two-stage Least Square (2SLS)

$$Y = \beta_0 + \beta_1 X + e \tag{I.6.1}$$

$$X = \gamma Z + \epsilon \tag{I.6.2}$$

6.1 assessing whether the instrumental variable Z is strong. Tip: remember *the rule of thumb*.

The first step is to make sure Z is a valid instrumental variable. A common *rule of thumb* is examine if the *F-statistics* of $Z > 10$ in isolating the part of X that is uncorrelated with e by regressing X on Z using OLS, with further expanded of *I.6.2*.

from `model.tvalues[1]` we get $t - stats_z$: `15.45862148821751`

In the case here, we possess only a single instrument, then F-statistic is equivalent to the squared value of the t-statistic associated with the instrument's coefficient in the first stage where $F - stats_z = 15.46^2$, meaning the instrumental variable Z is strong.

6.2 Estimate $\hat{X} = E[X \mid Z]$ according with the model in *Equation I.6.2*. Make sure to write down the estimated model parameters, p-values, etc. in a summary table.

Based on *Equation I.6.1*, we presents the following after regress X_i on Z_i with no intercept and yield a predicted $\hat{X}_i$.

	Parameters	T-Statistics	P-Values	Std. Err.
γ	0.49847622	15.48731022	1.19046284e-48	0.03218611

TABLE 6.1 Summary for X_i, Z_i First-stage Model with Equation I.6.2

6.3 Fitting the model in *Equation I.6.1*, but with the estimated values in *Exercise 6.2*.

Based on *Equation I.6.1*, we presents the following after replacing X_i with $\hat{X}_i$ with the regress Y on $\hat{X}_i$ with OLS.

	Parameters	T-Statistics	P-Values	Std. Err.
β_0	0.92185613	18.90980383	2.11844846e-68	0.04875017
β_1	0.66503933	6.74165978	2.64525083e-11	0.09864623

TABLE 6.2 Summary for Y_i, $\hat{X}_i$ Second-stage Model with Equation I.6.1

6.4 Derive the expression for the *reduced model*, fit such a model, and estimate β_1 as a function of the estimated coefficients in the reduced model and the model linking the treatment with the instrumental variable. Compare the estimate in this case, with that obtained using the *2SLS* approach.

To derive the reduced model and link the treatment with the instrumental variable, we can describe it as:

$$y = \theta_0 + \theta_1 z + w \qquad \text{(I.6.3)}$$

Here our reduced model is represent by the our `m3` model with `m3.params[1]`:

from `m3 = sm.OLS(Y,Zi).fit()` we get our $\hat{\theta}_1$ is `0.3315062938722091.`

This relates of *Y to Z and X to Z.*It is shown that we can actually use the ratio of our reduced model in our case since we only have one endogenous covariate.

$$\hat{\beta}_1^{\,2SLS} = \frac{\hat{\theta}_1}{\hat{\gamma}_1} \qquad \text{(I.6.3-1)}$$

Here our reduced model and model represent by our `m and m3` model:

from `m3.params[1]/m.params[0]` we get our estimate of $\hat{\beta}_1^{\,2SLS}$:

$$\hat{\beta}_1^{\,2SLS} = \frac{\hat{\theta}_1}{\hat{\gamma}_1} = \frac{0.3315062938722091}{0.4984762202721605} \qquad \text{(I.6.3-2)}$$

$$\hat{\beta}_1^{\,2SLS} = 0.665039334657151 \qquad \text{(I.6.3-3)}$$

Now let us take a look back to *Exercise 6.3*. We see in $\hat{X}_i$ the coefficient is 0.66503933, which matches our estimate of β_1 using the reduced model.

7 Appendix

```
import warnings
import math as m
import numpy as np
import scipy as sp
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
import linearmodels as lm
import statsmodels.api as sm
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf

from scipy import stats
from functools import reduce
from scipy.stats import bootstrap
from sklearn.utils import resample

from sklearn.decomposition import PCA
from statsmodels.tsa.ar_model import AutoReg
from sklearn.metrics import mean_squared_error
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedShuffleSplit
from statsmodels.stats.outliers_influence import variance_inflation_factor

%matplotlib inline
%config InlineBackend.figure_format = 'retina'

warnings.filterwarnings("ignore")
mpl.rcParams.update(mpl.rcParamsDefault)

```

```
#1.1
duncan = sm.datasets.get_rdataset("Duncan", "carData")

df = pd.DataFrame(data=duncan.data)
df['type'] = df['type'].astype('category') #change "type" to categorical

plt.rcParams.update({
    'text.usetex': True,
    'font.family': 'serif',
})
fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=2, ncols=2)

x1 = list(df['income'])
x2 = list(df['education'])
x3 = list(df['prestige'])

colors=['blue']
names=['income']
ax0.hist([x1], bins = 10, color=colors,label=names,edgecolor='black')
ax0.legend(prop={'size': 10})
ax0.set_xlabel('Individuals earning$3500 or more (%)')

colors=['green']
names=['education']
ax1.hist([x2], bins = 10, color=colors,label=names,edgecolor='black')
ax1.legend(prop={'size': 10})
ax1.set_xlabel('High-school Graduates (%)')

colors=['orange']
names=[ 'prestige']
ax2.hist([x3], bins = 10, color=colors,label=names,edgecolor='black')
ax2.legend(prop={'size': 10})
ax2.set_xlabel('Raters in a national survey who rated the
                occupation as good or better in prestige (%)')

```

```
#1.2
columns = ['income', 'education', 'prestige']
sns.pairplot(df.loc[:, columns], height=2)

#1.3
sns.set(style="whitegrid")
fig, ax = plt.subplots(1, 3, sharey=True, figsize=(12,4))

t1 = df[df['type'] == 'prof']
t2 = df[df['type'] == 'bc']
t3 = df[df['type'] == 'wc']

sns.violinplot(x=t1["type"], y=t1['income'], ax=ax[0]).set(xlabel=None)
sns.violinplot(x=t2["type"], y=t2['income'], ax=ax[0]).set(xlabel=None)
sns.violinplot(x=t3["type"], y=t3['income'], ax=ax[0]).set(xlabel=None)

sns.violinplot(x=t1["type"], y=t1['education'], ax=ax[1]).set(xlabel=None)
sns.violinplot(x=t2["type"], y=t2['education'], ax=ax[1]).set(xlabel=None)
sns.violinplot(x=t3["type"], y=t3['education'], ax=ax[1]).set(xlabel=None)

sns.violinplot(x=t1["type"], y=t1['prestige'], ax=ax[2]).set(xlabel=None)
sns.violinplot(x=t2["type"], y=t2['prestige'], ax=ax[2]).set(xlabel=None)
sns.violinplot(x=t3["type"], y=t3['prestige'], ax=ax[2]).set(xlabel=None)

#1.4 - 1.6
model = smf.ols("prestige ~ income + education", data = df).fit()
fit.summary()

Y = df['prestige']
residuals = Y - model.predict()
plt.hist(residuals, bins = 10, color='orange', edgecolor='black')

df['popf_bin'] = np.where(df['type'] == 'prof',1,0)
df

```

```
105
106  #2.1
107  df['popf_bin'] = np.where(df['type'] == 'prof',1,0)
108
109  #2.2 - 2.3
110  glm0 = smf.glm(formula = "popf_bin ~ income + education + prestige",
111                 data = df,
112                 family=sm.families.Binomial()).fit()
113
114  #2.4
115  glm1 = smf.glm(formula = "popf_bin ~ income", data = df, family=sm.families.Binomial()).fit()
116  glm2 = smf.glm(formula = "popf_bin ~ education", data = df, family=sm.families.Binomial()).fit()
117  glm3 = smf.glm(formula = "popf_bin ~ prestige", data = df, family=sm.families.Binomial()).fit()
118
119  #2.5
120  glm0.bic
121  glm1.bic
122  glm2.bic
123  glm3.bic
124
125  #3.1
126  np.random.seed(102030)
127  mu = np.array([5, 2, 1])
128  cov = np.array([[1, 0.4, 0], [0.4, 1.5, 0], [0, 0, 2]])
129  N = 20
130  x1, x2, x3 = np.random.multivariate_normal(mu, cov, N).T
131
132  #3.2
133  b = 5
134  c = 3
135
136  x_1 = x1
137  x_2 = np.dot(b, x2)
138  x_3 = np.dot(c, x3)
139  epsilon = np.random.normal(size=20)
140
```

```
141 ans=0
142 for i, j, k in zip(x_1, x_2, x_3):
143     ans = 10 + i + j + k + epsilon
144 
145 #3.3
146 B = 1000
147 N = 20
148 
149 bootstrap_data = []
150 
151 for b in range(B):
152 
153     indices = np.random.choice(N, N, replace=True)
154 
155     x1_bs, x2_bs, ans_bs = resample(x1, x2, ans, replace=True, n_samples=20)
156 
157     x_1 = x1_bs[indices]
158     x_2 = x2_bs[indices]
159     y_b = ans_bs[indices]
160     bootstrap_data.append([x_1,x_2,y_b])
161 
162 #3.4
163 from sklearn.linear_model import LinearRegression
164 
165 def boot_fn2(x1,x2,y):
166 
167     X = np.stack((x1, x2), axis=1)
168     m = LinearRegression(fit_intercept=True).fit(X,y)
169 
170     return m.intercept_, m.coef_
171 
172 coef_list = []
173 for b in range(B):
174 
175     coef_list.append(boot_fn2(bootstrap_data[b][0],bootstrap_data[b][1],bootstrap_data[b][2]))
176 
```

```
176
177   b_0 = []
178   b_1 = []
179   b_2 = []
180
181   for b in range(B):
182
183       b_0.append(coef_list[b][0])
184       b_1.append(coef_list[b][1][0])
185       b_2.append(coef_list[b][1][1])
186
187   #3.5
188   b0_me = np.array(b_0).mean()
189   b0_std = np.array(b_0).std()
190
191   b1_me = np.array(b_1).mean()
192   b1_std = np.array(b_1).std()
193
194   b2_me = np.array(b_2).mean()
195   b2_std = np.array(b_2).std()
196
197   fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=2, ncols=2)
198
199   colors=['blue']
200   names=['Beta_0']
201   ax0.hist([b_0], bins = 10, color=colors,edgecolor='black')
202   ax0.legend(prop={'size': 10})
203   ax0.set_xlabel(r'$\beta_{0}$')
204
205   colors=['green']
206   names=['Beta_1']
207   ax1.hist([b_1], bins = 10, color=colors,edgecolor='black')
208   ax1.legend(prop={'size': 10})
209   ax1.set_xlabel(r'$\beta_{1}$')
210
```

```
210
211   colors=['orange']
212   names=['Beta_2']
213   ax2.hist([b_2], bins = 10, color=colors,edgecolor='black')
214   ax2.legend(prop={'size': 10})
215   ax2.set_xlabel(r'$\beta_{2}$')
216
217   #3.6
218   np.random.seed(102030)
219   mu = np.array([5, 2, 1])
220   cov = np.array([[1, 0.4, 0], [0.4, 1.5, 0], [0, 0, 2]])
221   N = 1000
222   x11, x22, x33 = np.random.multivariate_normal(mu, cov, N).T
223
224   b = 5
225   c = 3
226
227   x101 = x11
228   x202 = np.dot(b, x22)
229   x303 = np.dot(c, x33)
230   epsilon = np.random.normal(size=1000)
231
232   anss = 0
233
234   for i, j, k in zip(x101, x202, x303):
235
236       anss = 10 + i + j + k + epsilon
237
238   #3.7
239   B = 1000
240   N = 20
241
242   bootstrap_data2 = []
```

```
242  bootstrap_data2 = []
243
244  for b in range(B):
245
246      indices = np.random.choice(N, N, replace=True)
247
248      x1_bs, x2_bs, ans_bs = resample(x101, x202, anss, replace=True, n_samples=1000)
249
250      xx1 = x1_bs[indices]
251      xx2 = x2_bs[indices]
252      yyb = ans_bs[indices]
253      bootstrap_data2.append([xx1,xx2,yyb])
254
255  coef_list2 = []
256  for b in range(B):
257
258      coef_list2.append(boot_fn3(bootstrap_data2[b][0],bootstrap_data2[b][1],bootstrap_data2[b][2]))
259
260  b_00 = []
261  b_11 = []
262  b_22 = []
263
264  for b in range(B):
265
266      b_00.append(coef_list2[b][0])
267      b_11.append(coef_list2[b][1][0])
268      b_22.append(coef_list2[b][1][1])
269
270  b00_me = np.array(b_00).mean()
271  b00_std = np.array(b_00).std()
272
273  b11_me = np.array(b_11).mean()
274  b11_std = np.array(b_11).std()
275
276  b22_me = np.array(b_22).mean()
277  b22_std = np.array(b_22).std()
```

```
279  fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=2, ncols=2)
280
281  colors=['blue']
282  names=['Beta_0']
283  ax0.hist([b_00], bins = 10, color=colors,edgecolor='black')
284  ax0.legend(prop={'size': 10})
285  ax0.set_xlabel(r'$\beta_{0}$')
286
287  colors=['green']
288  names=['Beta_1']
289  ax1.hist([b_11], bins = 10, color=colors,edgecolor='black')
290  ax1.legend(prop={'size': 10})
291  ax1.set_xlabel(r'$\beta_{1}$')
292
293  colors=['orange']
294  names=['Beta_2']
295  ax2.hist([b_22], bins = 10, color=colors,edgecolor='black')
296  ax2.legend(prop={'size': 10})
297  ax2.set_xlabel(r'$\beta_{2}$')
298
299  #3.8
300  x1x2 = []
301
302  for i, j in zip(x1, x2):
303      x1x2.append((i+j)/2)
304
305  data = {'x1': x1,
306          'x2': x2,
307          'x3': x1x2,
308          'y': ans,
309          }
310
311  df = pd.DataFrame(data)
312
313  x = df[['x1','x2','x3']]
314  y = df['y']
```

```
316  x = sm.add_constant(x)
317
318  model3 = sm.OLS(y, x).fit()
319
320  #3.9
321  x1x2 = []
322
323  for i, j in zip(x1, x2):
324      x1x2.append((i*j))
325
326  data2 = {'x1': x1,
327          'x2': x2,
328          'x3': x1x2,
329          'log(y)': np.log(ans)}
330
331  df2 = pd.DataFrame(data2)
332
333  xx = df2[['x1','x2','x3']]
334  yy = df2['log(y)']
335
336  xx = sm.add_constant(xx)
337
338  model33 = sm.OLS(yy, xx).fit()
339
340  vif1 = pd.DataFrame()
341  vif1["VIF Score"] = [variance_inflation_factor(x.values, i) for i in range(x.shape[1])]
342  vif1["Coefficient"] = x.columns
343
344  vif2 = pd.DataFrame()
345  vif2["VIF Score"] = [variance_inflation_factor(xx.values, i) for i in range(xx.shape[1])]
346  vif2["Coefficient"] = xx.columns
347
348  #4.1
349  nile = sm.datasets.get_rdataset("Nile").data
350  nile.set_index('time')
```

```
year = ('1880', '1900', '1920', '1940', '1960')
plt.figure(figsize = (10,4))
plt.plot(nile.time, nile.value)
plt.title('Nile River Annual Flow', fontsize = 20)
plt.ylabel('Value', fontsize = 16)
plt.xlabel('Time', fontsize = 16)
plt.axvline(x = 1871, color = 'k', linestyle = '--')
plt.axvline(x = 1880, color = 'k', linestyle = '--')
plt.axvline(x = 1900, color = 'k', linestyle = '--')
plt.axvline(x = 1920, color = 'k', linestyle = '--')
plt.axvline(x = 1940, color = 'k', linestyle = '--')
plt.axvline(x = 1960, color = 'k', linestyle = '--')
plt.axvline(x = 1970, color = 'k', linestyle = '--')

#4.2 - 4.3
fig, ax = plt.subplots(1,2,figsize=(20,5))
sm.graphics.tsa.plot_acf(nile["value"], lags=30, ax=ax[0])
sm.graphics.tsa.plot_pacf(nile["value"], lags=30, ax=ax[1])

#4.4
nile_train = nile[nile['time'].between(1871, 1965)]
nile_test = nile[nile['time'].between(1966, 1970)]

ar_1 = AutoReg(nile_train.value, lags = 1).fit()
ar_2 = AutoReg(nile_train.value, lags = 11).fit()

#4.5
predictions2 = ar_2.predict(start = nile_test.index[0],
                            end = nile_test.index[-1])

residuals2 = nile_test.value - predictions2

x_ticks = [95, 96, 97, 98, 99]
x_labels = ['1966', '1967', '1968', '1969', '1970']

plt.figure(figsize = (10,4))
```

```
387  plt.figure(figsize = (10,4))
388  plt.plot(nile_test.value)
389  plt.plot(predictions2)
390  plt.legend(('Data','Predictions'), fontsize = 15)
391  plt.title('Annual River Flow', fontsize = 20)
392  plt.ylabel('Value', fontsize = 15)
393  plt.xticks(ticks=x_ticks, labels=x_labels)
394
395  np.sqrt(np.mean(residuals2**2))
396
397  #5.1
398  df = pd.read_csv('/Users/niallj.632/Desktop/BI/2B Forcasting/dataset.csv')
399  df_var = df.filter(like = 'var')
400
401  X, y = df_var, df.target
402  sss = StratifiedShuffleSplit(n_splits = 5, test_size = 0.3, random_state = 42)
403
404  for train_index, test_index in sss.split(X, y):
405      X_train, X_test = X.iloc[train_index], X.iloc[test_index]
406      y_train, y_test = y.iloc[train_index], y.iloc[test_index]
407
408  log_reg = sm.Logit(y_train, X_train).fit()
409  y_pred = log_reg.predict(X_test)
410
411  false_positive_rate, true_positive_rate, threshold = roc_curve(y_test, y_pred)
412  roc_auc_score(y_test, y_pred)
413
414  plt.subplots(1, figsize=(10,10))
415  plt.plot(false_positive_rate, true_positive_rate)
416  plt.ylabel('True Positive Rate')
417  plt.xlabel('False Positive Rate')
418
419  #5.2
420  X, y = df_var, df.target
421  lr = LogisticRegression(penalty='none', random_state = 42, fit_intercept = True)
422  cvs = cross_val_score(lr, X, y, scoring="roc_auc", cv = 5)
```

```
424  mean_score = cvs.mean()
425  std_score = cvs.std()
426
427  #5.3
428  pca20 = PCA(n_components=20)
429  pca50 = PCA(n_components=50)
430  pca100 = PCA(n_components=100)
431
432  pca_features20 = pca20.fit_transform(X)
433  pca_features50 = pca50.fit_transform(X)
434  pca_features100 = pca100.fit_transform(X)
435
436  sns.set()
437
438  pca20.fit_transform(X)
439  plt.bar(
440      range(1,len(pca20.explained_variance_)+1),
441      pca20.explained_variance_
442      )
443  plt.xlabel(r'PCA, $p=20$')
444  plt.ylabel('Explained variance')
445
446  pca50.fit_transform(X)
447  plt.bar(
448      range(1,len(pca50.explained_variance_)+1),
449      pca50.explained_variance_
450      )
451  plt.xlabel(r'PCA, $p=50$')
452  plt.ylabel('Explained variance')
453
454  pca100.fit_transform(X)
455  plt.bar(
456      range(1,len(pca100.explained_variance_)+1),
457      pca100.explained_variance_
458      )
459  plt.xlabel(r'PCA, $p=100$')
```

```
460   plt.ylabel('Explained variance')
461
462   #5.4 - 5.5
463   p20 = pca20.fit_transform(X)
464   p50 = pca50.fit_transform(X)
465   p100 = pca100.fit_transform(X)
466
467   cvs20 = cross_val_score(lr, p20, y, scoring="roc_auc", cv = 5)
468   cvs50 = cross_val_score(lr, p50, y, scoring="roc_auc", cv = 5)
469   cvs100 = cross_val_score(lr, p100, y, scoring="roc_auc", cv = 5)
470
471   mean_score = cvs20.mean()
472   mean_score = cvs50.mean()
473   mean_score = cvs100.mean()
474
475   std_score = cvs20.std()
476   std_score = cvs50.std()
477   std_score = cvs100.std()
478
479   #6.1
480   sls = np.load('2sls.npy')
481   Y = sls[0]
482   X = sls[1]
483   Z = sls[2]
484
485   m = sm.OLS(X,Z).fit()
486
487   #6.2
488   m.params
489   m.pvalues
490   m.tvalues
491   m.bse
492
493   #6.3
494   xhat = m.predict(Z)
495   xh = sm.add_constant(xhat)
```

```
492
493   #6.3
494   xhat = m.predict(Z)
495   xh = sm.add_constant(xhat)
496
497   m2 = sm.OLS(Y,xh).fit()
498
499   m2.params
500   m2.pvalues
501   m2.tvalues
502   m2.bse
503
504   #6.4
505   Zi = sm.add_constant(Z)
506
507   m3 = sm.OLS(Y,Zi).fit()
508
509   m3.params[1]
510   m.params[0]
511   m3.params[1] / m.params[0]
512
```