Introduction to Statistical Learning with Applications

CM2: Multiple linear regression

Pedro L. C. Rodrigues

IN OUR PREVIOUS EPISODE...

We have a set of predictors and observations

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

IN OUR PREVIOUS EPISODE...

We have a set of predictors and observations

Example: California housing dataset

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

→ Each row represents the data for one district

Median house value for California districts

Median house age in district

Median population in district

Latitude at the center of the district

IN OUR PREVIOUS EPISODE...

We have a set of predictors and observations and want to find a relation between them.

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \longrightarrow f\Big(\begin{bmatrix} X_1 & X_2 & \dots & X_p \end{bmatrix}\Big) \approx Y$$

The objective function measures discrepancy between prediction and observations.

$$\underset{f \in \mathcal{L}^2(\mathbb{R}^p)}{\text{minimize}} \; \mathbb{E}_{Y, X_1, \dots, X_p}\left[\left\| Y - f\big(\begin{bmatrix} X_1 & X_2 & \dots & X_p \end{bmatrix}\big) \right\|^2 \right]$$ (mean squared error)

The solution can be obtained analytically... but in terms of an unknown quantity.

$$f^\star(x) = \mathbb{E}_{Y|X}[Y \mid X = x] = \int y \, \underline{p_{Y|X=x}(y)} \, \mathrm{d}y$$

IN OUR PREVIOUS EPISODE...

If we assume that the data follows a linear model as in

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon \quad \text{with} \quad \mathbb{E}[\varepsilon] = 0 \quad \mathrm{Var}(\varepsilon) = \sigma^2$$

Then we have that

$$p_{Y|X=x}(y) = p_\varepsilon\left(y - (\beta_0 + \sum_{i=1}^{p} \beta_i x_i)\right)$$

Therefore,

$$f^\star(x) = \mathbb{E}_{Y|X}\left[Y \mid X = x\right] = \beta_0 + \sum_{i=1}^{p} \beta_i x_i$$

Important: we made no assumptions about the specific shape of the pdf for the noise!

- Recap on Gaussian multiple linear regression
- Inference on the estimated parameters
- Some important remarks
- Categorical variables

The multiple linear regression model

We will delve further into the multiple linear regression model with p predictors

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon$$

It is very important to really understand each part of this model

The intercept can be interpreted as $\beta_0 = \mathbb{E}_{Y|X}\left[Y \mid X_1 = \cdots = X_p = 0\right]$

The β_i (with i > 0) should be interpreted as the average effect on Y of an unit increase in X_i while holding all other predictors fixed

Quantity ε is a zero-mean random error term independent of $X_1, \ldots, X_p$

The multiple linear regression model

When considering N observed data points from the multiple linear regression

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i$$

we often rewrite things with matrix notation so to facilitate the maths afterwards

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N \qquad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N1} & \dots & x_{Np} \end{bmatrix} \in \mathbb{R}^{N\times(p+1)}$$

$$\varepsilon = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{bmatrix} \in \mathbb{R}^N \qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} \in \mathbb{R}^{p+1}$$

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon$$

Estimating parameters from data

Q: How do we estimate the parameters from the observations in X and y ?

As we saw last week, a natural loss function to minimize is the MSE

$$\mathrm{MSE}(\beta) = \mathbb{E}_{XY}\left[\left(Y - \left(\beta_0 + \sum_{i=1}^{p} \beta_i X_i\right)\right)^2\right]$$

which can be approximated by

$$\mathrm{MSE}(\beta) \approx \frac{1}{N}(\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

Estimating parameters from data

Q: How do we estimate the parameters from the observations in X and y ?

Define the loss function
$$\begin{aligned}\mathcal{L}(\beta) &= (\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) \\ &= \mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\beta + \beta^\top(\mathbf{X}^\top\mathbf{X})\beta\end{aligned}$$

so that its minimizer is given by

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^{p+1}}{\operatorname{argmin}} \ \mathcal{L}(\beta) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Note that the predictions can be written as $\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \underbrace{\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T}_{\text{projection matrix}}\mathbf{y}$

Example: Consider the Advertising dataset available [here]

Y The observed variable is the number of sales of a particular product in different markets (e.g. different cities, neighbourhoods, etc.)

X_i The predictors are the advertising budget for the product on three different media: TV, radio, and newspaper.

We will assume that a multiple linear regression model can describe the data as per

$$\texttt{sales} = \beta_0 + \beta_1\ \texttt{TV} + \beta_2\ \texttt{radio} + \beta_3\ \texttt{newspaper} + \varepsilon$$

Using python we can estimate a multiple linear regression model

Example: Consider the Advertising dataset available [here]

CM2.py > ...

```python
import pandas as pd
import statsmodels.api as sm
import numpy as np

# load the dataset
filename = 'Advertising.csv'
df = pd.read_csv(filename, index_col=0)

# choose the predictors
X = df[['TV', 'radio', 'newspaper']]
X['intercept'] = 1 # add columns of ones

# choose the observed variable
y = df['sales']

# fit the multiple linear regression model
model = sm.OLS(y, X)
results = model.fit()

# print the summary of results
results.summary()
```

```
In [18]: print(results.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     570.3
Date:                Fri, 27 Dec 2024   Prob (F-statistic):           1.58e-96
Time:                        14:15:46   Log-Likelihood:                -386.18
No. Observations:                 200   AIC:                             780.4
Df Residuals:                     196   BIC:                             793.6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
TV             0.0458      0.001     32.809      0.000       0.043       0.049
radio          0.1885      0.009     21.893      0.000       0.172       0.206
newspaper     -0.0010      0.006     -0.177      0.860      -0.013       0.011
intercept      2.9389      0.312      9.422      0.000       2.324       3.554
==============================================================================
Omnibus:                       60.414   Durbin-Watson:                   2.084
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              151.241
Skew:                          -1.327   Prob(JB):                     1.44e-33
Kurtosis:                       6.332   Cond. No.                         454.
==============================================================================
```

Example: Consider the Advertising dataset available [here]

CM2.py > ...

```python
import pandas as pd
import statsmodels.api as sm
import numpy as np

# load the dataset
filename = 'Advertising.csv'
df = pd.read_csv(filename, index_col=0)

# choose the predictors
X = df[['TV', 'radio', 'newspaper']]
X['intercept'] = 1 # add columns of ones

# choose the observed variable
y = df['sales']

# fit the multiple linear regression model
model = sm.OLS(y, X)
results = model.fit()

# print the summary of results
results.summary()
```

NumPy

The fundamental package for scientific computing with Python

pandas

Fast, powerful, flexible data analysis and manipulation tool

statsmodels

Statistical models, hypothesis tests, and data exploration

Why not scikit-learn ? 😢

- Simple and efficient tools for **predictive** data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license – Made in France 🐓FR💪

However, no built-in way of doing statistical inference on the parameters, i.e. *p*-values

LinearRegression

class **sklearn.linear_model.LinearRegression**(*, *fit_intercept=True*, *copy_X=True*, *n_jobs=None*, *positive=False*) [source]

Ordinary least squares Linear Regression.

LinearRegression fits a linear model with coefficients w = (w1, ..., wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Example: Consider the Advertising dataset available **[here]**

CM2.py > ...

```python
import pandas as pd
import statsmodels.api as sm
import numpy as np

# load the dataset
filename = 'Advertising.csv'
df = pd.read_csv(filename, index_col=0)

# choose the predictors
X = df[['TV', 'radio', 'newspaper']]
X['intercept'] = 1 # add columns of ones

# choose the observed variable
y = df['sales']

# fit the multiple linear regression model
model = sm.OLS(y, X)
results = model.fit()

# print the summary of results
results.summary()
```

```
In [18]: print(results.summary())
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     570.3
Date:                Fri, 27 Dec 2024   Prob (F-statistic):           1.58e-96
Time:                        14:15:46   Log-Likelihood:                -386.18
No. Observations:                 200   AIC:                             780.4
Df Residuals:                     196   BIC:                             793.6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
TV             0.0458      0.001     32.809      0.000       0.043       0.049
radio          0.1885      0.009     21.893      0.000       0.172       0.206
newspaper     -0.0010      0.006     -0.177      0.860      -0.013       0.011
intercept      2.9389      0.312      9.422      0.000       2.324       3.554
==============================================================================
Omnibus:                       60.414   Durbin-Watson:                   2.084
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              151.241
Skew:                          -1.327   Prob(JB):                     1.44e-33
Kurtosis:                       6.332   Cond. No.                         454.
==============================================================================
```

The coefficients for `TV` and `newspaper` are very small, but are they **statistically significant**?

- Recap on Gaussian multiple linear regression
- Inference on the estimated parameters
- Some important remarks
- Categorical variables

Statistical inference on the parameters

To do statistical inference on each parameter we need a statistical model

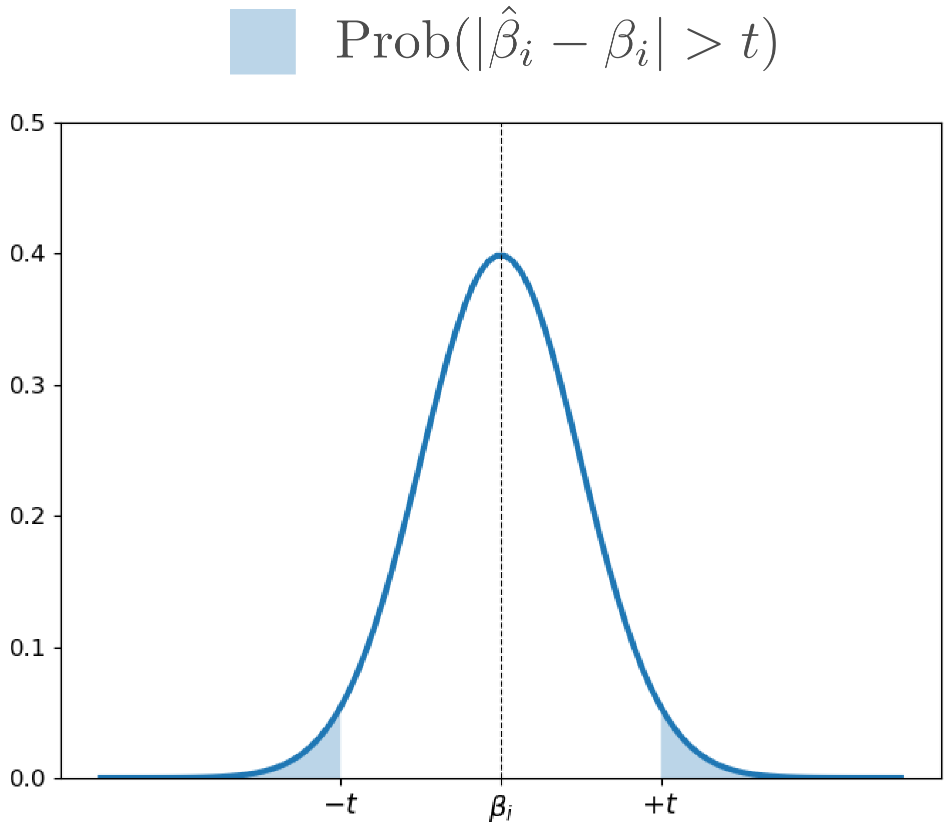
$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon \quad \text{with} \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

In this case we have that $\hat{\beta} \sim \mathcal{N}(\beta, \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1})$

and $$\frac{\hat{\beta}_i - \beta_i}{\sqrt{\sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}_{i+1,i+1}}} \sim \mathcal{N}(0,1)$$

We can now write a **statistical hypothesis test**

$$\mathcal{H}_0 : \quad \hat{\beta}_i = 0 \quad \text{vs} \quad \mathcal{H}_1 : \quad \hat{\beta}_i \neq 0$$



One-slide reminder of hypothesis testing

We assume the null hypothesis is valid and check whether the data push us to **reject it**

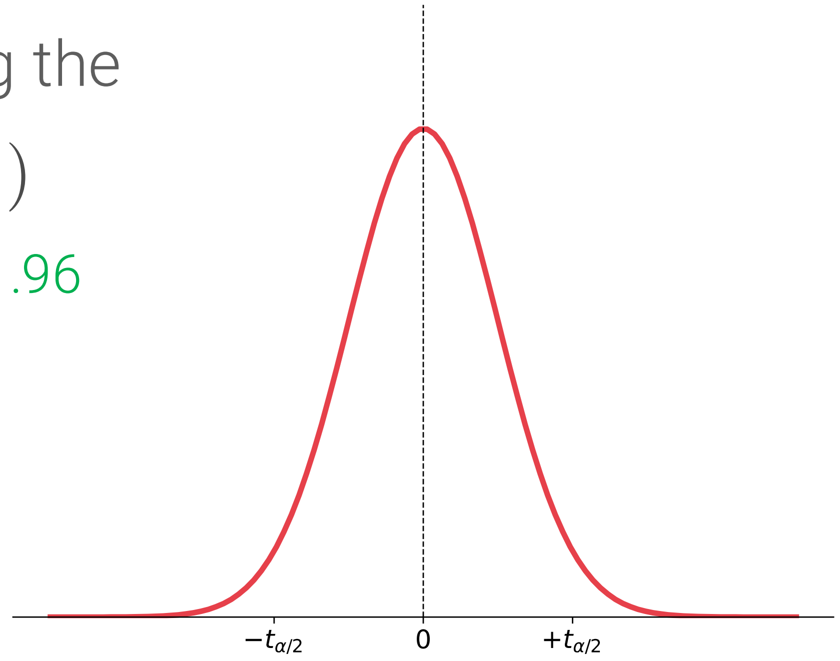
- Calculate the **test statistic** from the data: $\hat{T}_i = \dfrac{\hat{\beta}_i - 0}{\sqrt{\sigma^2 (\mathbf{X}^T\mathbf{X})^{-1}_{i+1,i+1}}}$

- Calculate the **probability** of the test statistic assuming the null hypothesis is valid $p_i = \mathrm{Prob}(|\hat{T}_i| > t_{\alpha/2} \mid \mathcal{H}_0)$

  ↳ for $\alpha = 0.05$ we have $t_{\alpha/2}=1.96$

- If $p_i < \alpha$ then reject null hypothesis:

  "there is very little evidence that the parameter you've estimated should correspond to a model with $\beta_i = 0$"



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.897
Model:                            OLS   Adj. R-squared:                  0.896
Method:                 Least Squares   F-statistic:                     570.3
Date:                Fri, 27 Dec 2024   Prob (F-statistic):           1.58e-96
Time:                        14:15:46   Log-Likelihood:                -386.18
No. Observations:                 200   AIC:                             780.4
Df Residuals:                     196   BIC:                             793.6
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
TV             0.0458      0.001     32.809      0.000       0.043       0.049
radio          0.1885      0.009     21.893      0.000       0.172       0.206
newspaper     -0.0010      0.006     -0.177      0.860      -0.013       0.011
intercept      2.9389      0.312      9.422      0.000       2.324       3.554
==============================================================================
Omnibus:                       60.414   Durbin-Watson:                   2.084
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              151.241
Skew:                          -1.327   Prob(JB):                     1.44e-33
Kurtosis:                       6.332   Cond. No.                         454.
==============================================================================
```

Statistical inference on the parameters

We usually only have an **estimator** for the variance of the Gaussian white noise

$$\hat{\sigma}^2 = \frac{1}{N}\|\mathbf{y} - \mathbf{X}\hat{\beta}\|^2 \qquad \mathbb{E}[\hat{\sigma}^2] = \frac{N-(P+1)}{N}\,\sigma^2$$

The test statistic follows rather a t-student distribution with n-p-1 degrees of freedom

$$\frac{\hat{\beta}_i - \beta_i}{\sqrt{\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}_{i+1,i+1}}} \Longrightarrow \frac{\hat{\beta}_i}{\hat{\mathrm{se}}[\hat{\beta}_i]} \sim t_{N-p-1} \qquad \hat{\mathrm{se}}[\hat{\beta}_i] = \sqrt{\hat{\sigma}^2(\mathbf{X}^T\mathbf{X})^{-1}_{i+1,i+1}}$$

Coming back to the example:

Small but statistically significant →

	coef	std err	t	P>|t|
TV	0.0458	0.001	32.809	0.000
radio	0.1885	0.009	21.893	0.000
newspaper	-0.0010	0.006	-0.177	0.860
intercept	2.9389	0.312	9.422	0.000

Statistical inference on the parameters

Q: What, exactly, is `statsmodels` testing?

The hypothesis being tested is:

"Y is a linear function of all the X_i (with i between 1 and p), with constant variance, independent Gaussian white noise, and is just so happens that $\beta_i = 0$"

Remember that $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Important: This means that whether $\beta_i = 0$ is true or not can depend on which other variables are included in the regression!

Statistical inference on the parameters

```
# choose the predictors
X = df[['TV', 'radio', 'newspaper']]
X['intercept'] = 1 # add columns of ones
```

	coef	std err	t	P>|t|
TV	0.0458	0.001	32.809	0.000
radio	0.1885	0.009	21.893	0.000
newspaper	-0.0010	0.006	-0.177	0.860
intercept	2.9389	0.312	9.422	0.000

```
# choose the predictors
X = df[['newspaper']]
X['intercept'] = 1 # add columns of ones
```

	coef	std err	t	P>|t|
newspaper	0.0547	0.017	3.300	0.001
intercept	12.3514	0.621	19.876	0.000

The parameter **becomes** statistically significant

Testing for multiple coefficients

It is often useful to also test for the significance of a **group** of coefficients

$\hat{\sigma}^2_{\text{null}}$ $$\mathcal{H}_0 \;:\; Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_q X_q + \underbrace{0X_{q+1} + \cdots + 0X_p}_{\text{p-q coefficients are all zero}} + \varepsilon$$

$\hat{\sigma}^2_{\text{full}}$ $$\mathcal{H}_1 \;:\; Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_q X_q + \beta_{q+1} X_{q+1} + \cdots + \beta_p X_p + \varepsilon$$

We compare the estimated MSE for each model with a F-test

$$\frac{\hat{\sigma}^2_{\text{null}} - \hat{\sigma}^2_{\text{full}}}{\hat{\sigma}^2_{\text{full}}} \times \frac{1/(p-q)}{1/(N-p-1)} \sim F_{p-q,N-p-1}$$

"Does letting the slopes for X_{q+1}, ..., X_p be non-zero reduce the MSE more than we would expect just by noise?"

Testing for multiple coefficients

An important special case is to test if **all coefficients are zero** or not.

$\hat{\sigma}^2_{\text{null}}$ $$\mathcal{H}_0 \;:\; Y = \beta_0 + 0X_1 + \cdots + 0X_p + \varepsilon$$

$\hat{\sigma}^2_{\text{full}}$ $$\mathcal{H}_1 \;:\; Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_q X_q + \beta_{q+1} X_{q+1} + \cdots + \beta_p X_p + \varepsilon$$

The test statistic becomes then

$$\frac{s_Y^2 - \hat{\sigma}^2_{\text{full}}}{\hat{\sigma}^2_{\text{full}}} \times \frac{1/p}{1/(N-p-1)} \sim F_{p,N-p-1}$$

which is often called the test of significance of the whole regression.

Example: Consider the mtcars dataset – we want to predict **mpg**

```
Description:

     The data was extracted from the 1974 _Motor Trend_ US magazine,
     and comprises fuel consumption and 10 aspects of automobile design
     and performance for 32 automobiles (1973-74 models).

Format:

     A data frame with 32 observations on 11 variables.

       [, 1]  mpg   Miles/(US) gallon
       [, 2]  cyl   Number of cylinders
       [, 3]  disp  Displacement (cu.in.)
       [, 4]  hp    Gross horsepower
       [, 5]  drat  Rear axle ratio
       [, 6]  wt    Weight (lb/1000)
       [, 7]  qsec  1/4 mile time
       [, 8]  vs    V/S
       [, 9]  am    Transmission (0 = automatic, 1 = manual)
       [,10]  gear  Number of forward gears
       [,11]  carb  Number of carburetors
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.869
Model:                            OLS   Adj. R-squared:                  0.807
Method:                 Least Squares   F-statistic:                     13.93
Date:                Fri, 27 Dec 2024   Prob (F-statistic):           3.79e-07
Time:                        15:09:11   Log-Likelihood:                -69.855
No. Observations:                  32   AIC:                             161.7
Df Residuals:                      21   BIC:                             177.8
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
cyl           -0.1114      1.045     -0.107      0.916      -2.285       2.062
disp           0.0133      0.018      0.747      0.463      -0.024       0.050
hp            -0.0215      0.022     -0.987      0.335      -0.067       0.024
drat           0.7871      1.635      0.481      0.635      -2.614       4.188
wt            -3.7153      1.894     -1.961      0.063      -7.655       0.224
qsec           0.8210      0.731      1.123      0.274      -0.699       2.341
vs             0.3178      2.105      0.151      0.881      -4.059       4.694
am             2.5202      2.057      1.225      0.234      -1.757       6.797
gear           0.6554      1.493      0.439      0.665      -2.450       3.761
carb          -0.1994      0.829     -0.241      0.812      -1.923       1.524
intercept     12.3034     18.718      0.657      0.518     -26.623      51.229
==============================================================================
```

... but the F-test does

None of the t-tests reject H_0

- Recap on Gaussian multiple linear regression
- Inference on the estimated parameters
- Some important remarks
- Categorical variables

Q: Why multiple regression isn't a bunch of simple regressions?

"The slopes we get for each variable in multiple regression are not the same as if we did p separate simple regression. Why not?"

Suppose the true model of the data is $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$

If we did a simple regression of Y just on X_1 we would estimate the slope as per

$$\frac{\mathrm{Cov}(X_1, Y)}{\mathrm{Var}(X_1)} = \beta_1 + \beta_2 \frac{\mathrm{Cov}(X_1, X_2)}{\mathrm{Var}(X_1)}$$

indirect contribution through correlation with X_2

X_1's direct contribution to Y

X_2's contribution to Y

When X_1 and X_2 are correlated, we can predict a bit of X_1 from X_2 and vice-versa

Multicollinearity

Remember the expression for the parameters of the multiple linear regression model

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

We have always assumed $\mathbf{X}^T\mathbf{X}$ is invertible. What happens if it is not the case?

- The variance of the estimated parameters $\mathrm{Var}(\hat{\beta}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$ will blow up!

  (And even if it's not exactly singular but almost, the variances will be very big)

- The test statistic is $\dfrac{\hat{\beta}_i - \beta_i}{\sigma^2(\mathbf{X}^T\mathbf{X})^{-1}_{i+1,i+1}} \downarrow$ so $\mathbf{H_0}$ will very often not be rejected

Let's see an example...

OLS Regression Results

Dep. Variable:	mpg	R-squared:	0.869
Model:	OLS	Adj. R-squared:	0.807
Method:	Least Squares	F-statistic:	13.93
Date:	Fri, 27 Dec 2024	Prob (F-statistic):	3.79e-07
Time:	15:09:11	Log-Likelihood:	-69.855
No. Observations:	32	AIC:	161.7
Df Residuals:	21	BIC:	177.8
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
cyl	-0.1114	1.045	-0.107	0.916	-2.285	2.062
disp	0.0133	0.018	0.747	0.463	-0.024	0.050
hp	-0.0215	0.022	-0.987	0.335	-0.067	0.024
drat	0.7871	1.635	0.481	0.635	-2.614	4.188
wt	-3.7153	1.894	-1.961	0.063	-7.655	0.224
qsec	0.8210	0.731	1.123	0.274	-0.699	2.341
vs	0.3178	2.105	0.151	0.881	-4.059	4.694
am	2.5202	2.057	1.225	0.234	-1.757	6.797
gear	0.6554	1.493	0.439	0.665	-2.450	3.761
carb	-0.1994	0.829	-0.241	0.812	-1.923	1.524
intercept	12.3034	18.718	0.657	0.518	-26.623	51.229

None of the t-tests reject H_0

- Recap on Gaussian multiple linear regression
- Inference on the estimated parameters
- Some important remarks
- Categorical variables

Handling categorical predictors

In some cases, we might want to regress Y using also **qualitative** predictors:

- "How does the salary of an employee relates with its gender?"
- "Can the nationality of a person help predict his/her life expectancy?"

These are examples where the levels of the predictor have no notion of ordering

Binary categories

(One-hot encoding)

- Pick one of two levels as the **reference** or baseline category.
- Add column X_B to the design matrix X for each data point indicating whether it belongs to baseline ($X_B = 1$) or not ($X_B = 0$)
- Regress on $Y = \beta_0 + \beta_B X_B + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon$

Dummy variable encoding categories

Continuous predictors

Handling categorical predictors: the binary case

Consider having one continuous predictor and one binary categorial predictor

$$Y = \beta_0 + \beta_B X_B + \beta_1 X_1 + \varepsilon$$

- We have that the coefficient for the categorial predictors is

$$\beta_B = \mathbb{E}[Y \mid B = 1, X_1 = x] - \mathbb{E}[Y \mid B = 0, X_1 = x]$$

"It's the expected difference in the response between members of the reference category and members of the other category, all else being equal"

- There are basically two models with different intercepts:

$$Y = \beta_0 + \beta_1 X_1 \qquad\qquad Y = (\beta_0 + \beta_B) + \beta_1 X_1$$

model for the baseline category

model for the other category

Handling categorical predictors: the binary case

Example: We want to regress the **effective life** of an industrial tool before it needs maintenance based on two predictors:

- How much stress we impose to the tool (i.e. rotation speed)
- Its brand (either A or B)

$$\texttt{life} = \beta_0 + \beta_{\texttt{brand}}\ X_{\texttt{brand}} + \beta_{\texttt{rpm}}\ X_{\texttt{rpm}} + \varepsilon$$

$$\texttt{life} = (\beta_0 + \beta_{\texttt{brand}}) + \beta_{\texttt{rpm}}\ \texttt{rpm} + \varepsilon$$

model for brand B

$$\texttt{life} = \beta_0 + \beta_{\texttt{rpm}}\ \texttt{rpm} + \varepsilon$$

model for brand A

Handling categorical predictors

```
In [85]: df
Out[85]:
```

	life	rpm	brand
0	18.73	610	A
1	14.52	950	A
2	17.43	720	A
3	14.54	840	A
4	13.44	980	A
5	24.39	530	A
6	13.34	680	A
7	22.71	540	A
8	12.68	890	A
9	19.32	730	A
10	30.16	670	B
11	27.09	770	B
12	25.40	880	B
13	26.05	1000	B
14	33.49	760	B
15	35.62	590	B
16	26.07	910	B
17	36.78	650	B
18	34.95	810	B
19	43.67	500	B

```python
import pandas as pd
import statsmodels.api as sm
import numpy as np

# load the dataset
filename = 'effectivelife.csv'
df = pd.read_csv(filename, index_col=0)
df['brand'] = df['brand'].astype("category")

# encode the categorical features
df_enc = pd.get_dummies(
    df, dtype=np.float64, drop_first=True)

# choose the predictors
X = df_enc.drop(columns=['life'])
X['intercept'] = 1 # add columns of ones

# choose the observed variable
y = df_enc['life']

# fit the multiple linear regression model
model = sm.OLS(y, X)
results = model.fit()

# print the summary of results
print(results.summary())
```

	coef	std err	t	P>|t|	[0.025	0.975]
rpm	-0.0266	0.005	-5.887	0.000	-0.036	-0.017
brand_B	15.0043	1.360	11.035	0.000	12.136	17.873
intercept	36.9856	3.510	10.536	0.000	29.579	44.392

Handling categorical predictors

Q: Why not add two columns to the design matrix? (one for each level)

A: The two columns would be **linearly dependent** (they will always add up to one) so the data would end up being collinear → problems with inversion.

Q: Why not two slopes? (one for each level)

A: This is perfectly reasonable, but would require a different kind of linear model using **interactions** between predictors. We won't see this in this course.

Handling categorical predictors

If our categorical predictor has more than just two levels, we can simply

- Pick one of the k levels as the **reference** or baseline category.
- Add k-1 columns to the design matrix X which are **indicators** for the other categories.
- Regress as usual, getting k-1 **constrasts** for the categorical predictors

In our previous example, if we had three levels for the brand (A, B, or C) we would get

$$\begin{aligned}
\texttt{life} &= \beta_0 + \beta_{\texttt{brand}=B} X_{\texttt{brand}=B} + \beta_{\texttt{brand}=C} X_{\texttt{brand}=C} + \beta_{\texttt{rpm}} X_{\texttt{rpm}} + \varepsilon \\
\texttt{life} &= \beta_0 + \beta_{\texttt{rpm}} X_{\texttt{rpm}} + \varepsilon \longrightarrow \text{if brand A} \\
\texttt{life} &= (\beta_0 + \beta_{\texttt{brand}=B}) + \beta_{\texttt{rpm}} X_{\texttt{rpm}} + \varepsilon \longrightarrow \text{if brand B} \\
\texttt{life} &= (\beta_0 + \beta_{\texttt{brand}=C}) + \beta_{\texttt{rpm}} X_{\texttt{rpm}} + \varepsilon \longrightarrow \text{if brand C}
\end{aligned}$$

Handling categorical predictors

But what if we have **too many** categories? This would make the data matrix too big!

One idea, would have been to try using an ordinal encoder from scikit-learn`

OrdinalEncoder

```
class sklearn.preprocessing.OrdinalEncoder(*, categories='auto', dtype=<class 'numpy.float64'>, handle_unknown='error', unknown_value=None, encoded_missing_value=nan, min_frequency=None, max_categories=None)
```

[source]

Encode categorical features as an integer array.

The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are converted to ordinal integers. This results in a single column of integers (0 to n_categories - 1) per feature.

Handling categorical predictors

But what if we have **too many** categories? This would make the data matrix too big!

Another possibility would be to use a target encoder from `scikit-learn` too.

TargetEncoder

class **sklearn.preprocessing.TargetEncoder**(*categories='auto'*, *target_type='auto'*, *smooth='auto'*, *cv=5*, *shuffle=True*, *random_state=None*)

[source]

Target Encoder for regression and classification targets.

Each category is encoded based on a shrunk estimate of the average target values for observations belonging to the category. The encoding scheme mixes the global target mean with the target mean conditioned on the value of the category (see [MIC]).

Handling categorical predictors

But what if we have **too many** categories? This would make the data matrix too big!

Let's consider an example with a `winereviews` dataset

```
In [30]: df
Out[30]:
       country                                        description  ...                variety                    winery
0           US  This tremendous 100% varietal wine hails from ...  ...     Cabernet Sauvignon                     Heitz
1        Spain  Ripe aromas of fig, blackberry and cassis are ...  ...         Tinta de Toro  Bodega Carmen Rodríguez
2           US  Mac Watson honors the memory of a wine once ma...  ...        Sauvignon Blanc                  Macauley
3           US  This spent 20 months in 30% new French oak, an...  ...             Pinot Noir                     Ponzi
4       France  This is the top wine from La Bégude, named aft...  ...    Provence red blend    Domaine de la Bégude
...        ...                                                ...  ...                    ...                       ...
150925   Italy  Many people feel Fiano represents southern Ita...  ...           White Blend    Feudi di San Gregorio
150926  France  Offers an intriguing nose with ginger, lime an...  ...       Champagne Blend                 H.Germain
150927   Italy  This classic example comes from a cru vineyard...  ...           White Blend                 Terredora
150928  France  A perfect salmon shade, with scents of peaches...  ...       Champagne Blend                    Gosset
150929   Italy  More Pinot Grigios should taste like this. A r...  ...           Pinot Grigio              Alois Lageder

[150930 rows x 10 columns]
```

We want to predict the points columns (values between 80 and 100) based on the other predictors from the dataframe: 6 categorical and 1 numerical

Handling categorical predictors

But what if we have **too many** categories? This would make the data matrix too big!

Let's consider an example with a `winereviews` dataset

```
numerical_features = ["price"]
categorical_features = [
    "country",
    "province",
    "region_1",
    "region_2",
    "variety",
    "winery",
]
target_name = "points"
```

```
In [31]: print(n_unique_categories.sort_values(ascending=False))
    ...:
winery      14810
region_1     1236
variety       632
province      455
country        48
region_2       18
dtype: int64
```

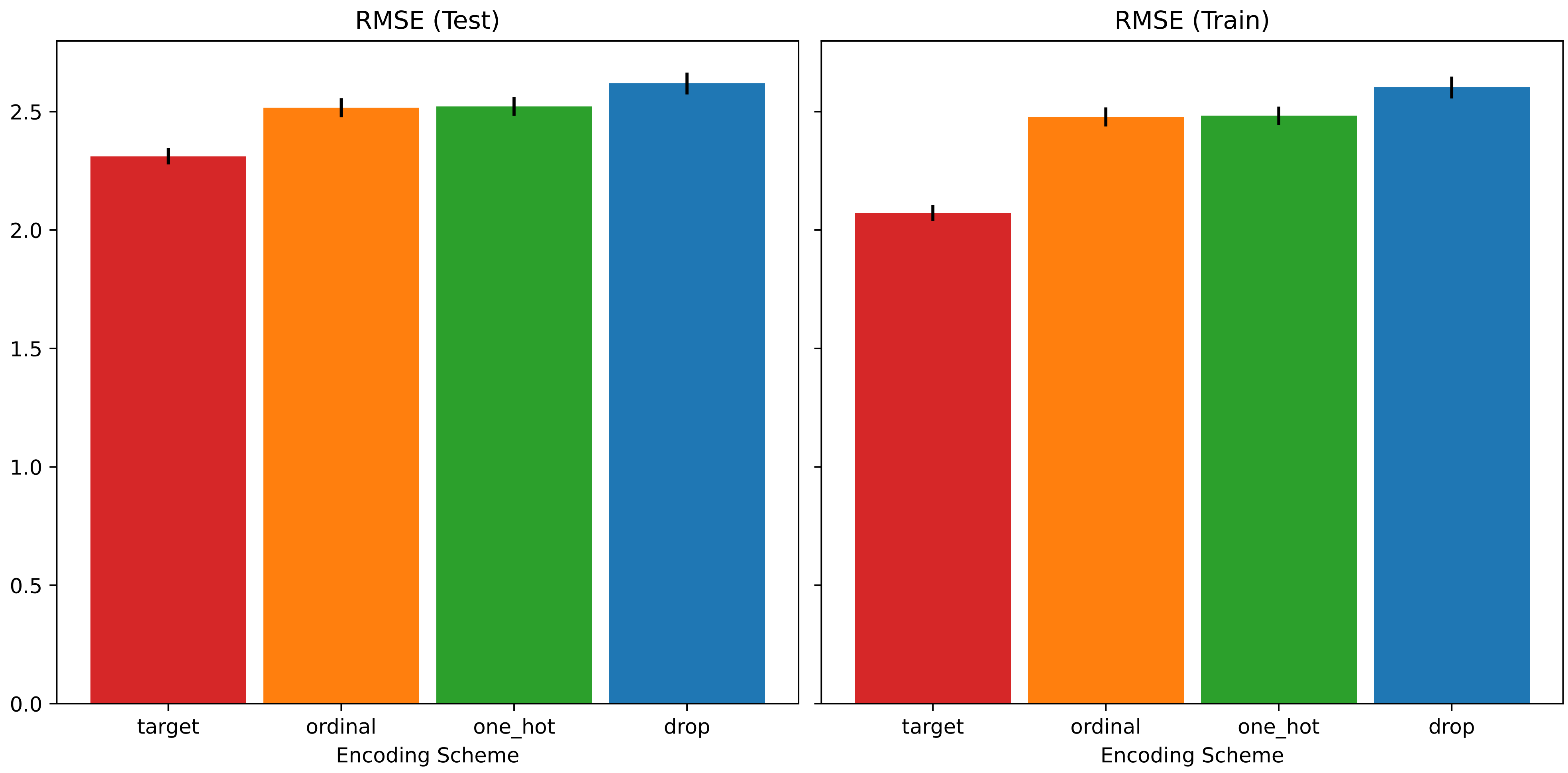
Let's compare the performance of **regression** with different encoding schemes.

Forget about inference for now, focus on predictive power

Handling categorical predictors

But what if we have **too many** categories? This would make the data matrix too big!

Let's consider an example with a `winereviews` dataset



Before going to the TP rooms…

If you're using your own computer, please be sure to install all the packages necessary for our class.

main 2025-main / requirements.txt

plcrodrigues adding materials

Code Blame 10 lines (10 loc) · 89 Bytes

```
numpy
pandas
scikit-learn
matplotlib
geopandas
geodatasets
skrub
jupyter
ipykernel
igraph
```

Just follow these three steps…

1) Install conda

2) Run conda create -n isla2026 python=3.12

3) Run pip install -r requirements.txt