

---

TP 2 : Linear regression

---

For this lab, you have to upload a single **ipynb** file. Please use the following script to format your filename (bad name will lead to a 1 point penalty):

```
# Change here using YOUR own first and last names
fn1 = "john"
ln1 = "smith"
filename = "_".join(map(lambda s: s.strip().lower(),
                        ["SD-TSIA204_lab2", ln1, fn1])) + ".ipynb"
```

You have to upload it on eCampus before Wednesday 16/02/2022, 23h59 in the folder corresponding to your lab group. Out of 20 points, 5 are specifically dedicated to:

- Presentation quality: writing, clarity, no typos, visual efforts for graphs, titles, legend, colorblindness, etc. (2 points).
- Coding quality: indentation, PEP8 Style, readability, adapted comments, brevity (2 points)
- No bug on the grader's machine (1 point)

**Note:** you can use [https://github.com/agramfort/check\\_notebook](https://github.com/agramfort/check_notebook) to check your notebook is fine, and also use <https://github.com/kenkoooo/jupyter-autopep8> to enforce pep8 style.

**Beware:** labs submitted late, by email or uploaded in a wrong group folder will be graded 0/20.

---

We are given a dataset; we are trying to predict the value of variable  $Y$  with some covariates  $X$  (as columns) for which there are  $n$  i.i.d. measurements (as the rows of the data provided) over  $p$  covariates. However, it is not clear whether all the covariates are relevant for the prediction of  $Y$ . In this TP, we are going to consider several variants of the OLS to make a regressor under this setting and identify relevant variables.

1) Preprocess the data:

- Set the random seed to 0.
- Load the data. Print the mean, and standard deviation of every covariate. Is the data centered? Normalized? Standardized?
- Separate the data frame in two matrices,  $X$  and  $Y$ , containing the input and output data respectively.
- Separate the covariate data in train and test sets. Save one fourth of the data as testing.
- Center and standardize the train and test data and plot its mean and variance again. Use the functions `fit_transform` and `transform`.
- Create two empty dataFrames of names `df_test` and `df_coef`.

2) Write a function to compute the determination coefficient and another to compute the mean squared error.

3) Using sklearn utilities, fit a linear regression model on the train set that we will use as baseline.

- apply the linear regression of the sklearn library to the train data,
- print the determination coefficient and the MSE of the test data,

- add a column named `OLS` to the `df_test` dataframe that contains the predicted values for the sample and
- add a column named `OLS` to the `df_coef` dataframe that contains the estimated coefficients.

Note: adding a column of name `my_col` with the values of the list `my_list` is done as follows:  
`df['my_col'] = my_col.`

### PCA

- 4) Plot a heatmap of the covariance matrix. Compute the singular value decomposition of the covariance matrix. For consistency in the notation use  $U, s, V = \text{SVD}(X^\top X)$ .
- 5) Show that the  $i$ -th singular value of  $X$ ,  $\sigma_i$ , and the  $i$ -th eigenvalue of the covariance matrix  $(n - 1)^{-1} X^\top X$ ,  $\lambda_i$ , are related as follows  $\lambda_i = (n - 1)^{-1} \sigma_i^2$
- 6) In PCA we transform the data to a new coordinate system such that the greatest variance by some scalar projection of the data lies on the first coordinate (called the first principal component, PC1), the second greatest variance in the second PC and so on. The PCs are computed given the above SVD, as  $XU$ . Instead of using the whole transformation,  $XU$ , we will use (as an approximation) the first 2 PCs, i.e., the first 2 columns in  $XU$ . Which is the variance explained by each component? Plot the projection of the data into the 1st and 2nd components.
- 7) In a biplot (as the one in the slides) we find the projected sample (as in the previous question) and the loadings of the variables (as arrows from the center). They are defined as the product of the singular vectors  $U$  and the square root of the singular values. Explain what the loadings are in terms of the variance and plot them for the 1st and 2nd directions.
- 8) We will apply the method "PCA before OLS", which consists in applying OLS with output  $Y$  and input  $X \cdot U[:, 60]$ , where  $U[:, 60]$  contains the eigenvectors (associated with the 2 largest eigenvalues) of the covariance matrix. Run linear regression as follows:
  - Compute projected data for both the train and the test data,
  - apply the linear regression of the sklearn library to the low dimensional data,
  - print the determination coefficient and the MSE of the test data and
  - add a column named `pca_ols` to the `df_test` dataframe that contains the predicted values for the test sample.

### Regularization

- 9) Using sklearn utilities, fit a Lasso model on the train set.
  - apply the `LassoCV` of the sklearn library: it uses cross-validation internally for different values of the regularization parameter `alpha`: try 30 different values for `alpha`, spaced evenly on a log scale between  $1e - 3$  and  $1e1$  (see function `np.logspace`),
  - print the determination coefficient and the MSE of the test data,
  - print the regularization parameter that the algorithm selects,
  - add a column named `lasso` to the `df_test` dataframe that contains the predicted values for the sample and
  - add a column named `lasso` to the `df_coef` dataframe that contains the estimated coefficients.
- 10) Using sklearn utilities, fit a Ridge model on the train set.
  - apply the `RidgeCV` of the sklearn library: it uses crossvalidation internally for different values of the regularization parameter `alpha`: try 30 different values for `alpha`, spaced evenly on a log scale between  $1e - 1$  and  $10e2$  (see function `np.logspace`).
  - print the determination coefficient and the MSE of the test data,
  - print the regularization parameter that the algorithm selects,
  - add a column named `Ridge` to the `df_test` dataframe that contains the predicted values for the sample and

- add a column named `Ridge` to the `df_coef` dataframe that contains the estimated coefficients.

### Variable selection

- 11) Program the method of the forward variable selection. You can use the test statistics of the test for nullity (as seen during the course). For the moment, do not define the stop criterion for the method, i.e. add a variables at each time until all the variables are used. Provide the order of the variable selection.
- 12) Stop criterion: We choose to stop if the  $p$ -value is larger than 0.05. Illustrate the method providing (i) the 3 graphs of the test statistics obtained when selecting the 1st, 2nd and 3rd variables (in abscissa: the indices of the variables; in the ordinate: the value of the test statics), (ii) the graphs of the first 50  $p$ -values (each associated to a selected variable). On the same plot, trace the horizontal line with the ordinate 0.05. Finally, provide the list of the selected variables.
- 13) Run OLS on the selected variables.
  - apply the OLS of the `sklearn` library.
  - print the determination coefficient and the MSE of the test data,
  - add a column named `FVS` to the `df_test` dataframe that contains the predicted values for the sample and
  - add a column named `FVS` to the `df_coef` dataframe that contains the estimated coefficients.

### Comparison

- 14) Summarize the results of all the methods: Compare the values of the coefficients from the different methods by plotting `df_coef`. Avoid using lines (i.e., use only markers) for a better interpretability of the plot.
- 15) How many coefficients are set exactly to zero by each of the methods considered in the data frame `df_coef`? (That is: OLS, Lasso, Ridge and FVS but not PCA before OLS). Why is the number of coefficients set to 0 so different for Lasso and Ridge?
- 16) Give a pairplot of the `df_test` dataframe. Let the fitted coefficient of the previous problems be  $\hat{\theta}^d$  for the ridge,  $\hat{\theta}^{ols}$  for the OLS,  $\hat{\theta}^{pca}$  for the PCA before OLS,  $\hat{\theta}^{ls}$  for the lasso. Specify the equation to predict the value of a new, unseen data point  $x$ .

### Confidence intervals and bootstrapping

- 17) Load the Diabetes dataset form sklearn (`load_diabetes(return_X_y=True)`) and let  $X$  be the 3rd column of the loaded data (the feature matrix consists only on variable 3 from the original dataset). We are going to compute approximately the confidence intervals (CI) for the  $\theta_1$  coefficient via Bootstrapping.
  - Give the OLS coefficient for the slope,  $\theta_1$ .
  - Code manually the expression for the confidence intervals and evaluate it for  $\alpha = 0.05$  (using `statsmodel` is not allowed).
  - We denote the concatenation of  $X$  and  $y$  as  $Xy$ . Generate  $k$  bootstrap samples from  $Xy$ , for `k=np.linspace(10,10000)`. A bootstrap sample has the same dimension as  $Xy$ , is formed of datapoints (rows) drawn, with replacement, from  $Xy$ .
  - (For each value of  $k$ ) (i) Estimate the  $\theta_1$  coefficient for each bootstrapped sample with `sklearn`, (ii) get an approximation of the CI by getting the  $(\alpha, 1 - \alpha)$  percentile of the values in step (i).
  - Plot, for each value of  $k$ , which is the difference between the approximated CI in the previous step and the real CI.
  - Does the CI increase with  $\alpha$ ? Why?