Contents

1 Introduction

Multiple linear regression is widely used in many applications to model linear relationships between a set of predictors and a response. One of the more common and simpler methods of fitting such a model is known as the ordinary least squares method (OLS), which is found to be a Best Linear Unbiased Estimator (BLUE) by the Gauss-Markov theorem, that is, it gives the unbiased estimates of each regression coefficient with the lowest possible variance, provided that assumptions are satisfied. Furthermore, it has a main advantage of it being easily interpretable as a parametric model as opposed to non-parametric models.

The motivation for this project arises from one of these assumptions, that is, the data matrix, $\boldsymbol{X}$, is full rank, i.e. none of the predictors are (nearly) perfectly correlated with each other. In cases where this is not true, OLS is unable to estimate regression parameters due to singularity. In such cases, there are several methods we can employ to address this issue, whether to drop one of the correlated variables, or to combine them, etc., based on the discretion of the data analyst. In this project, we investigate how two of the commonly used software for regression analysis, Python (`sklearn` and `statsmodel`) and R each handles perfect collinearity when fitting the model.

Furthermore, we embrace this opportunity to explore further on the topic of multicollinearity, such as its impact on model prediction and interpretation. We also apply several known methods of detecting collinearity in the data, whether through hypothesis testing, or by using measures such as the variance inflation factor (VIF) and condition numbers/indices. In addition to aforementioned methods of addressing collinearity, we consider shrinkage methods such as ridge regression and lasso to alleviate the impact of overfitting (due to collinearity) on variance at the cost of negligible bias. We also briefly explore another potential pitfall in regression analysis, that is confounding variables, by examining examples of Simpson's paradox.

Here, we lay down the general setup for fitting a linear regression model. Given an $n \times p$ data matrix $\boldsymbol{X}$ (n samples and p predictors), corresponding to an $n \times 1$ column matrix $\boldsymbol{y}$ containing the observed response, we model the relationship between p predictors, X_i and the response variable, Y as

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon,$$

where ε represents the random error. We fit the model to the data collected for the expected response $E[Y]$ by estimating each β_i using the least squares method to obtain $\hat{\beta}_i$ so that we

have

$$E[Y] = \hat{\beta}_0 + \sum_{i=1}^{p} \hat{\beta}_i X_i.$$

Finally, it should be noted that there are different conventions to what collinearity and multicollinearity mean according to varying sources. In this report, we will use these two terms interchangably to refer to the correlation between two or more predictors in a regression model.

2 Objectives

There are three main objectives for this project:

1. Explore issues that arise from singularity.
2. Understand singular value decomposition (SVD).
3. Investigate how software deals with singularity when building linear models.

In practical settings, it is unlikely to obtain a set of perfectly correlated data due to inevitable noise. However, even near perfect collinearity leads to singularity when attempting to fit the regression model using software. We hence aim to explore how a rank-deficient data matrix can lead to singularity and the issues that arise from it. We also explore pitfalls caused by moderate to high multicollinearity in terms of interpretation and prediction. Subsequently, we look into commonly used methods to handle situations where regressors are highly correlated depending on purpose of analysis.

The next objective is to undetstand how to apply the singular value decomposition in the computations of the least square estimates. The least squares method estimates the regression coefficients of a model with p predictors by fitting it to a dataset with n observations using the following equation

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{1}$$

where $\hat{\boldsymbol{\beta}}$ is a $p \times 1$ column matrix containing the least square estimates, $\boldsymbol{X}$ is a $n \times p$ data matrix, and $\boldsymbol{y}$ is a $n \times 1$ column matrix containing the observed response values. With the presence of perfect collinearity, the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is singular and hence $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ does not exist. However, the least squares solver of Python uses the Moore-Penrose inverse (or pseudoinverse) of $\boldsymbol{X}$, $\boldsymbol{X}^+$, which can be used to calculate the minimum norm approximation of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$, and can be computed using the SVD of $\boldsymbol{X}^T\boldsymbol{X}$ (see Appendix Section 7.3).

Lastly, it is observed that there exists discrepancy between how different software handles highly correlated predictors when fitting a multiple linear regression model via OLS. In this project, we aim to understand the reason behind this difference in behaviour by referring to standard documentation and examining program output in Python and R. As an extension, methods for detecting and addressing multicollinearity using software are also explored.

3 Methodology

We narrowed our focus to three least squares solvers of interests: Python's `scikit-learn` library, Python's `statsmodel` library, and R's `lm` function. In each case, we randomly generated several sets of artificial data using three regressors where only one pair is correlated, each with 100 samples. The use of artificial data has the advantage where the true relationship between the response and predictors is known and self-determined, and that the degree of correlation and noise in the data can be manually specified. Note that the distribution used in data generation is irrelevant for our topic of study.

The data ranges from 0 to 100 and is uniformly distributed. The degree of collinearity was then varied between perfect and moderate to observe differences in output. The pseudoinverse of the data matrix was also computed and multiplied with the generated response to give the theoretical minimum norm solution for the coefficient estimates. Then, these results were compared with those given by the least squares solver function to confirm whether these functions use the pseudoinverse of the data matrix to estimate the least squares coefficients as suggested by documentation.

The true underlying relationship to which the artificial data was generated was set to be

$$Y = X_1 + 2X_2 + 3X_3,$$

where X_1 and X_2 are correlated with $X_2 = 2X_1$. We then vary the amount of uniformly distributed noise added to different generated sets: from no noise (perfect collinearity) to some number sampled from intervals [-25,25], [-50,50], and [-100,100]. The train-test ratio chosen for splitting the dataset into training and testing sets is fixed at 80-20. Lastly, to evaluate prediction performance, metrics such as the (adjusted) multiple coefficient of determination R^2 and mean squared error were used.

The scaling effect of `scikit_learn`'s `StandardScaler` on the model was also given attention by referring to standard documentation as well as by testing it using artificially generated data. For data with and without high multicollinearity present respectively, we

fitted two models each: one with predictors scaled by `StandardScaler` and one without. The output was then compared.

We have also performed regression analysis on California housing and applied methods to address collinearity using `statsmodel`. Subsequently, ridge regression models and lasso models were also used to examine its effectiveness in decreasing the standard errors of these models.

3.1 Methods to detect collinearity

One simple way to detect collinearity is through a test for correlation, where we test the hypothesis

$$H_0 : \rho = 0 \qquad \text{vs.} \qquad H_1 : \rho \neq 0,$$

where ρ is the population correlation coefficient between a chosen pair of predictors. However, there are two main reasons to opt for an alternative method.

Firstly, as the number of predictors included in the model increases, the number of tests required to cover every combination of pairs increases. With large number of tests, the probability that we commit at least one error is high and hence we risk incorrectly identifying correlated pairs. More importantly, even if we find that every pair of predictors has no significant correlation, we cannot make the same conclusion about predictors amongst groups of other numbers.

3.1.1 Variance inflation factor (VIF)

A commonly used method for identifying highly correlated predictors is by using the variance inflation factor (VIF), given by

$$\text{VIF}_k = \frac{1}{1 - R_k^2},$$

where R_k^2 is the multiple coefficient of determination of the k^{th} predictor regressed against the other predictors, assuming that it has zero mean. Intuitively, a higher value of R_k^2 indicates a good fit, hence a strong correlation between the k^{th} regressor and some of the other regressors. Note that uncorrelated regressors would simply have a coefficient close to zero.

The motivation behind the formulation of the VIF is to calculate the ratio between the variance of a coefficient estimate if it was uncorrelated with the other regressors (baseline case), and the actual variance. For example, a coefficient estimate having a VIF of 4 corresponds to its variance being 4 times of what it would have been if it was uncorrelated with every other predictor. In other words, VIF measures the factor by which the variance of the

regression coefficients is inflated above what it would have been if $R_k^2 = 0$, hence its name.

There are different rule of thumbs for the thresholds beyond which one may consider the VIF value of a particular predictor to be high, ranging roughly from 4 to 10. There is no definitively correct answer, and we have no choice but to make decisions based on our interpretation of the VIF value. Once predictors with high VIF's are identified, we can then examine the correlation matrix to identify which the predictors which each high VIF predictor is correlated to.

3.1.2 Condition numbers

Another metric for measuring the degreee of collinearity is the condition number (condition indices) of the data matrix $\boldsymbol{X}$, which indicates how sensitive the response is to changes in $\boldsymbol{X}$. Aside from using it to determine whether a model is highly collinear, the condition number of $\boldsymbol{X}^T\boldsymbol{X}$ is used to check for a potential source of large numerical error due to collinearity when computing $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ using software. In this case, it indicates the multiplicative factor to which round-off errors are amplified due to computer arithmetic being performed to finite precision.

The formula for the condition number of a $m \times n$ matrix $\boldsymbol{A}$ with rank $r \leq n$, denoted $\kappa(\boldsymbol{A})$, is given by

$$\kappa(\boldsymbol{A}) = \frac{\sigma_1}{\sigma_r},$$

where σ_i are the singular values of $\boldsymbol{A}$ arranged in descending order. Note that definitions will differ for varying contexts, giving rise to different condition numbers. The point of discussion here is the condition number for inversion.

Similar to the case of VIF, there is no exact threshold for a condition number to be considered high, but a rule of thumb is any condition numbers exceeding 20 indicates the matrix to be ill-conditioned.

3.2 Methods to address collinearity

We present a few commonly-used methods to address moderate to perfect collinearity.

In the case of perfect collinearity, one of the correlated predictors is completely described by the other, hence our only choice is to remove one of them. This is however not recommended for non-perfect collinearity as we are at risk of committing omitted variable bias. In such cases, we may want to either combine some of the variables or use shrinkage

methods such as ridge regression or LASSO.

There are many possible ways of combining correlated variables, but in most cases the simple approach of adding the columns up (such as what R does) gives sufficiently improved results. Note that it is important that we consider the context when doing this, that is, whether combining the two variables make sense in context. An example is a model predicting the salary of employees based on predictors such as years of experience, gender, test scores for course A, and test scores for course B. Obviously, any employee who does well on one test is expected to do well in the other, and so we may consider adding the two test scores together to form a single column ”total test scores for course A and B”, then the results obtained may be interpreted accordingly.

For ridge regression and LASSO, otherwise known as L_2 and L_1 regularisation respectively, we find that it successfully decreases coefficient standard errors at the cost of increasing the bias (and hence decreasing R^2 score compared to the OLS model) by a negligible amount. However, in the face of perfect collinearity, ridge and lasso does not yield interpretable coefficients, where the correlated predictors have equally distributed coefficients due to the effect of standardising the predictors beforehand (see Appendix Section 7.2), as the ridge and LASSO coefficient estimates are not scale equivariant.

4 Results and discussion

4.1 Python's `scikit-learn` library

(include screenshot of results?)

- Evenly distributes weights amongst the correlated predictors (found out to be due to `StandardScaler`)
- Uses pseudoinverse
- No actual need to scale predictors for least squares regression (common practice in ML when algorithm involves gradient descent or when we perform principle component analysis)
- Does not give flags or warnings upon detection of high multicollinearity. Needs to be assessed manually by the data analyst.

4.2 Python's `statsmodel` library

(include screenshot of results?)

- Looks like it evenly distributes weights throughout the correlated predictors.
- Uses pseudoinverse
- Gives warning upon detect of high to perfect collinearity.

4.3 R's `lm` function

(include screenshot of results?)

- Looks like it dropped one collinear column and add the two coefficients together.
- Seems to combine collinear columns.
- Gives a warning when highly correlated variables are detected
- Option available to abort process upon detection of high mutlicollinearity.

4.4 Impact of multicollinearity

High multicollinearity mainly poses a major problem to model interpretation. Notably, it inflates the variance and covariances of least squares estimates and causes estimates and their variance to become sensitive to minor changes in data. As a result, we may obtain different results when performing variable selection after removing or adding samples from the dataset, or obtain incorrect signs of cofficients estimates. Another curious effect is that the t-values of correlated predictors may be insignificant yet removing these predictors from the model results in a lower R^2 value. Although this seems contradictory by intuition, it is actually reasonable behaviour, as the prediction is contributed by both variables while the t-values indicate the contribution of one variable after accounting for the others.

When two predictors have perfect correlation, we have what is known as perfect collinearity. In such a case, at least one of the columns of the data matrix $\boldsymbol{X}$ is a linear combination of the other columns. This causes the matrix $\boldsymbol{X}^T\boldsymbol{X}$ to be singular (see Appendix section 7.1), consequently causing $\hat{\boldsymbol{\beta}}$ unable to be computed (see Equation 1). However, `statsmodel`'s and `scikit_learn`'s least square solvers compute the pseudoinverse of $\boldsymbol{X}^T\boldsymbol{X}$ when estimating the regression coefficients, giving the least norm approximated solution (which is interpretable) when $\boldsymbol{X}$ is rank-deficient. However, the aforementioned impacts of high collinearity still apply hence it should be addressed.

It is important to note that collinearity does not, in any way, impact predictions or its precision. Hence, if the model is not expected to be interpreted, multicollinearity is no issue of concern and may be ignored. Moreover, even for more statistically-oriented

analysts, there may also not be a need to address high collinearity provided that it does not inflate standard errors to a problematic extent (see ”A caution regarding rules of thumb for variance inflation factors” by O’Brien), as there are many other factors which can suppress the inflation effect of standard errors, one of which is the sample size. Quoting the words of O’Brien, ”collinearity does not hurt as long as it does not bite”.

4.5 Confounding variables

In addition to the pitfall of collinearity in performing regression analysis, confounding variables, or confounders, can also severely distort our results if not given attention. The concept of confounders is illustrated using Simpson’s paradox, where the relationship between individual predictors and the response disappears or reverses when all predictors are included in a single model. The variables which cause such occurences are referred to as confounders.

We give the classic textbook example of the relationship between ice cream sales and shark attacks. We initially observe a positive correlation between these two variables, yet upon the inclusion of the confounder, temperature, we find that this correlation vanishes. One should avoid the confusion between confounders and correlated variables by noting that in addition to being correlated, confounders should also have a causal relationship to the variables involved in the confounding effect.

Although including confounders inherently introduces collinearity into the model, failing to account for them causes us to commit omitted variable bias in addition to causing the aforementioned confounding effects according to Simpson’s paradox. (We note the difference between these confounding effects and the effect of collinearity on the coefficient signs: the latter may still occur even after including confounders in the model due to high collinearity causing coefficient estimates to become sensitive to changes in observed data). Hence, by weighing each consequence, we generally always include confounders as we have well-established methods for effectively addressing collinearity. Unfortunately, it is more often the case that confounders are excluded unintentionally, as there is no standard method of identifying them. Furthermore, this is not a problem exclusive to regression modelling, but observational studies in general. Hence, we highlight this further in Section 6.

5 Conclusion

After examining Python’s and R’s program output, we have observed differences in how they each handle the issue of multicollinearity. In particular, R’s behaviour of combining

correlated predictors while giving a warning upon detection of high multicollinearity aligns best with standard practice. It is also found that in the process of estimating the regression coefficients, R's `lm`, Python's `statsmodel`, and `sklearn` calculates the pseudoinverse of $\boldsymbol{X}^T\boldsymbol{X}$ rather than $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ so that a result may be obtained even in the presence of singularity.

Additionally, we have observed a difference between the three's behaviour of flagging problematic multicollinearity, which indicates the general attitude of different fields that uses regression modelling. For example, `sklearn` aligns with machine learning engineers who are generally more concerned about making accurate predictions and hence would not be as concerned about the impacts of collinearity on model interpretability. This is in stark contrast to statisticians who bear greater interest in investigating the relationships between the regressors and the response. Hence, as opposed to `scikit_learn`, they would be more likely to use `statsmodel` or R, which in addition to flagging high multicollinearity, also affords a more extensive function library for the purpose of performing analysis on the model.

Taking these distinctions into account, it depends ultimately on the data analyst to make the final decision on which approach is best to handle anomalies in the model after performing analysis according to context and domain knowledge. Nevertheless, it is still crucial that one is made aware of these differences so as to select the appropriate tools depending on the individual's purpose and objective of regression modelling.

6 Future recommendations

1. Explore the same issue in other software commonly used in the industry such as (?)
2. Confounding and causality.
3. Partial effect, total effect, direct effect

A Singularity due to perfect collinearity

Let $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{u} & k\boldsymbol{u} & \boldsymbol{v} \end{pmatrix}$ where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ are arbitrarily linearly independent column vectors containing n observations of a particular feature with $\boldsymbol{u}^T = \begin{pmatrix} u_1 & \dots & u_n \end{pmatrix}$ and $\boldsymbol{v}^T = \begin{pmatrix} v_1 & \dots & v_n \end{pmatrix}$, with $k \in \mathbb{R}$ such that the column $k\boldsymbol{u}$ is a scalar multiple of the first column.

Consider

$$\begin{aligned}
\boldsymbol{X}^T\boldsymbol{X} &= \begin{pmatrix} \boldsymbol{u}^T \\ k\boldsymbol{u}^T \\ \boldsymbol{v}^T \end{pmatrix}\begin{pmatrix} \boldsymbol{u} & k\boldsymbol{u} & \boldsymbol{v} \end{pmatrix} \\
&= \begin{pmatrix} u_1 & \dots & u_n \\ ku_1 & \dots & ku_n \\ v_1 & \dots & v_n \end{pmatrix}\begin{pmatrix} u_1 & ku_1 & v_1 \\ \vdots & \vdots & \vdots \\ u_n & ku_n & v_n \end{pmatrix} \\
&= \sum_{i=1}^{n} \begin{pmatrix} u_i \\ ku_i \\ v_i \end{pmatrix}\begin{pmatrix} u_i & ku_i & v_i \end{pmatrix} \\
&= \sum_{i=1}^{n} \begin{pmatrix} u_i^2 & ku_i^2 & u_iv_i \\ ku_i^2 & k^2u_i^2 & ku_iv_i \\ u_iv_i & ku_iv_i & v_i^2 \end{pmatrix}.
\end{aligned}$$

The second row of $\boldsymbol{X}^T\boldsymbol{X}$ is k times the first row for every i. After summing up the rank-1 matrices, the second row of $\boldsymbol{X}^T\boldsymbol{X}$ will also be k times the first. So, $\boldsymbol{X}^T\boldsymbol{X}$ is not full rank, and hence is singular.

By writing matrix-matrix products in this form, each term in the sum is a rank-1 matrix. However, under no collinearity, the rows in every term will not necessarily be dependent on some other rows **by the same factor**. In the formulation above, the third row differs from the first row by a multiplicative factor of v_i/u_i (this is consistent with the case of collinear columns since $ku_i/u_i = k$), which is a constant under no collinearity. This fraction will differ for varying i, thus after adding each term up, there is no common factor between the two rows.

Alternatively for a simpler and less analytic approach, we look at each entry in the product matrix as multiplying rows with columns:

$$\boldsymbol{X}^T\boldsymbol{X} = \begin{pmatrix} \boldsymbol{u}\cdot\boldsymbol{u} & \boldsymbol{u}\cdot(k\boldsymbol{u}) & \boldsymbol{u}\cdot\boldsymbol{v} \\ (k\boldsymbol{u})\cdot\boldsymbol{u} & (k\boldsymbol{u})\cdot(k\boldsymbol{u}) & (k\boldsymbol{u})\cdot\boldsymbol{v} \\ \boldsymbol{v}\cdot\boldsymbol{u} & \boldsymbol{v}\cdot(k\boldsymbol{u}) & \boldsymbol{v}\cdot\boldsymbol{v} \end{pmatrix} = \begin{pmatrix} \boldsymbol{u}\cdot\boldsymbol{u} & k\boldsymbol{u}\cdot\boldsymbol{u} & \boldsymbol{u}\cdot\boldsymbol{v} \\ k(\boldsymbol{u}\cdot\boldsymbol{u}) & k(k\boldsymbol{u}\cdot\boldsymbol{u}) & k(\boldsymbol{u}\cdot\boldsymbol{v}) \\ \boldsymbol{v}\cdot\boldsymbol{u} & k(\boldsymbol{v}\cdot\boldsymbol{u}) & \boldsymbol{v}\cdot\boldsymbol{v} \end{pmatrix}.$$

We observe that the second row is k times the first row as expected. The third row again is not collinear with the first two. Suppose they are, then we solve for some $\lambda \in \mathbb{R}$ such that

$$\boldsymbol{v}\cdot\boldsymbol{u} = \lambda(\boldsymbol{u}\cdot\boldsymbol{u}) \quad \text{and} \quad \boldsymbol{v}\cdot\boldsymbol{v} = \lambda(\boldsymbol{u}\cdot\boldsymbol{v}).$$

It then follows that

$$\begin{aligned}
\|\boldsymbol{v}\|^2 &= \boldsymbol{v} \cdot \boldsymbol{v} \\
&= \lambda(\boldsymbol{v} \cdot \boldsymbol{u}) \\
&= \lambda^2(\boldsymbol{u} \cdot \boldsymbol{u}) \\
&= \lambda^2\|\boldsymbol{u}\|^2 \\
\|\boldsymbol{v}\| &= \lambda\|\boldsymbol{u}\| \\
&= \frac{\boldsymbol{v} \cdot \boldsymbol{u}}{\|\boldsymbol{u}\|^2}\|\boldsymbol{u}\| \\
\boldsymbol{v} &= \frac{\boldsymbol{v} \cdot \boldsymbol{u}}{\|\boldsymbol{u}\|^2}\boldsymbol{u}
\end{aligned}$$

i.e. $\boldsymbol{v}$ is its own orthogonal projection on vector $\boldsymbol{u}$, implying that they are collinear, which is a contradiction. Hence the third row is linearly independent of the first two rows.

Since the least squares coefficient estimates are given by

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

the singularity of $\boldsymbol{X}^T\boldsymbol{X}$ causes least squares to fail to obtain estimates for .

B Effect of `StandardScaler` on collinear columns in data matrix

From the documentation, `scikit_learn`'s `StandardScaler` calculates the Z-score of each entry by subtracting each entry by the mean of its column and dividing by the standard deviation of its column. Suppose that the true relationship between regressors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ is $\boldsymbol{x}_2 = k\boldsymbol{x}_1$. Then we have

$$\bar{x_2} = k\bar{x_1} \qquad \text{and} \qquad s_2^2 = k^2 s_1^2.$$

So for collinear columns $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the entries of scaled columns $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are

$$u_{i1} = \frac{x_{i1} - \bar{x_1}}{s_1} \qquad \text{and} \qquad u_{i2} = \frac{x_{i2} - \bar{x_2}}{s_2} = \frac{kx_{i1} - k\bar{x_1}}{ks_1} = \frac{x_{i1} - \bar{x_1}}{s_1} = u_{i1}.$$

C Minimum norm approximation using pseudoinverse

We can compute the pseudoinverse of $\boldsymbol{X}^T\boldsymbol{X}$ using its SVD as follows.

Let $\boldsymbol{A} = \boldsymbol{X}^T\boldsymbol{X}$. If the SVD of $\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^T$, then the pseudoinverse of $\boldsymbol{A}$, $\boldsymbol{A}^+$, is given by

$$\begin{aligned}
\boldsymbol{A}^+ &= ((\boldsymbol{U}\Sigma\boldsymbol{V}^T)^T()\boldsymbol{U}\Sigma\boldsymbol{V}^T)^{-1}(\boldsymbol{U}\Sigma\boldsymbol{V}^T)^T \\
&= (\boldsymbol{V}\Sigma\boldsymbol{U}^T\boldsymbol{U}\Sigma\boldsymbol{V}^T)^{-1}(\boldsymbol{U}\Sigma\boldsymbol{V}^T)^T \\
&= (\boldsymbol{V}\Sigma^2\boldsymbol{V}^T)^{-1}(\boldsymbol{V}\Sigma\boldsymbol{U}^T) \\
&= (\boldsymbol{V}^T)^{-1}(\Sigma^{-1})^2(\boldsymbol{V})^{-1}\boldsymbol{V}\Sigma\boldsymbol{U}^T \\
&= \boldsymbol{V}\Sigma^{-1}\boldsymbol{U}^T
\end{aligned}$$

Now, for $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$, if the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has solutions, then $\boldsymbol{x}^* = \boldsymbol{A}^+\boldsymbol{b}$ is an exact solution and has the smallest possible norm, i.e. $\|\boldsymbol{x}^*\| \leq \|\boldsymbol{x}\|$ for all $\boldsymbol{x}$.

The proof is as follows: Since $\boldsymbol{A}^+$ is a generalised inverse, $\boldsymbol{A}^+\boldsymbol{b}$ must be a solution to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Now, for any solution $\boldsymbol{x} \in \mathbb{R}^n$, consider its orthogonal decomposition via $\boldsymbol{A}^+\boldsymbol{A} \in \mathbb{R}^{n\times n}$:

$$\boldsymbol{x} = (\boldsymbol{A}^+\boldsymbol{A})\boldsymbol{x} + (\boldsymbol{I} - \boldsymbol{A}^+\boldsymbol{A})\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{b} + (\boldsymbol{I} - \boldsymbol{A}^+\boldsymbol{A})\boldsymbol{x}.$$

Then by the Pythagorean theorem, we have

$$\|\boldsymbol{x}\|^2 = \|\boldsymbol{A}^+\boldsymbol{b}\|^2 + \|(\boldsymbol{I} - \boldsymbol{A}^+\boldsymbol{A})\boldsymbol{x}\|^2 \geq \|\boldsymbol{A}^+\boldsymbol{b}\|^2.$$

Hence $\|\boldsymbol{x}\| \geq \|\boldsymbol{A}^+\boldsymbol{b}\|$.

References