Ordinary Least Squares and Ridge Regression of Franke Function and Boston Housing Dataset

Galapon, Arthur V., Phan Thi Dieu Trang

February 2021

Abstract

When it comes to predicting a quantitative response, a variety of regression methods have been considered. On the one hand, Ordinary Least Squares (OLS), a widely useful tool for estimating the parameters in a regression model and minimize the value of the sum of squared error, is present. On the other hand, there is a Ridge regression which is a well-known technique for shrinking the coefficient estimates towards zero resulting in a better fit. In this project, we first want to study these two methods by creating the random data from Franke's function combined with the stochastic noise and then apply them in Boston Housing data set aiming for the best prediction for the prices of the house represented by Median value of owner-occupied homes in $ 1000's variable.

1 Introduction

The term Machine Learning was arisen in 1959 since its techniques and algorithms allow a computer to learn from experience without explicitly programmed. During the last two decades, the applications of this science are significantly shown in many aspects such as traffic alerts, dynamic pricing, Google translate, social media and even solving high-dimensional differential equations or complicated quantum mechanical many-body problems. In particular, Machine Learning plays such a vital role in developing an accurate model to get a good prediction from the data and understand the relationship between the model and the actual data. Among different methods, OLS regression

analysis is the most often used to estimate the function for this model. This linear regression basically predicts a continuous dependent variable from a group of independent variables and find their correlations which is formed by a straight line showing the best estimate in all data points. The best model is obtained when the sum of the squares in the difference error between the observed and predicted values of the dependent variable gets a minimum. However, this model is penalized for its choice of weights leading to overfitting in high-dimensional data. Hence, Ridge regression, an alternative method which penalizes the model for the sum of squared value of the weights, is introduced. In this way, the weights not only tend to have smaller absolute values and more evenly distributed, but also tend to be close to zeros. Our objective is first to utilize Franke's function to study these two methods and combine them in turn with resampling techniques like the bootstrapping and cross validation. We then apply all the methods in Boston Housing data set to verify the best model resulting in a good prediction of the prices of the houses. Indeed, we find out that the one with polynomial degree two and Ridge regression has a best fit with our data.

2 Formalism

2.1 Model evaluation

In Machine Learning, there is a theorem called "No Free Lunch" [1]. It describes the phenomena that there is no single algorithm that is best suited for all possible scenarios and data sets. Hence, it is important to decide for any given set of data which method gives the best results. In order to evaluate the performance of a statistical learning method on a certain data set, we need to determine some parameters which reveal how well its predictions actually match the observed data. In the regression models, the most commonly-used measure is the mean squared error (MSE),

$$MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2, \tag{1}$$

where $\hat{y}_i$ is the prediction for the i^{th} observation and y_i is the corresponding true value. The squared component results in larger errors having larger penalties. Thus, if we want to make the predicted responses as close as possible to the true ones, it is essential to minimize the MSE.

The major goal in regression is to find an algorithm that not only fits well to our given data but also predicts accurately a future outcome. This is called generalizability of our algorithm. The question is how we "arrange" our data to allow us to verify if our algorithm is good for unseen data. For that, it is common to split our data in training and test set.

- Training set: Normally, the proportion of these data would be 80% which are used to develop feature sets, train our algorithms, tune hyperparameters, compare models and more other functions aiming to a final model.
- Test set: The rest 20% would belong to this set. After achieving the final model, these data are used to estimate an assessment of the model's performance.

The MSE in the above equation is computed using the training data that is used to fit the model, so it is called training MSE. In general, we do not really focus on how well the method works on the training data. Instead, we are interested in the prediction for the unseen test data. Therefore, we propose a parameter called test MSE which could be computed mathematically as the average squared prediction error for these test data, that is

$$MSE_{test} = Ave(y_i - \hat{y}_i)^2. \tag{2}$$

We desire to choose the model in which the test MSE is as small as possible so that it will be good at predicting. In some settings, the test data set is available so it is easy to use the Eq.2 to obtain the test MSE, thus achieve the optimal model which has the smallest test MSE. But in the case there is no test observation, can we rely on training MSE to pick the best model? The answer is not really. Since test and training MSE does not show the proportional relationship. In other words, a method with the lowest training MSE does not mean that its test MSE is minimum. Sometimes, they show the opposite trend which is illustrated in Fig.1 [8]. This figure shows typical relationships between model flexibility and both training and test MSE. In the left-hand panel, the black curve represents a known population model while the other curves are models that have been fit to the sample data. The green curve is the one matches the data very well but it shows a poor fit to the true model. The blue one seems to have a good estimation of the true model. It can be seen in the right-hand panel, as flexibility increases,

- the training MSE decrease,
- the test MSE initially declines, then, at some points, it levels off and starts to increase.

Depend on these trend, we can observe that the orange and the green curves have high test MSE while the blue one minimize this quantity. Hence, the model building the blue curve would be the optimal model. The tendency of test and training MSE in this case is a very typical behavior when working with several regression methods with varying degree of flexibility. And it is considered as the fundamental property of statistical learning that holds regardless of the particular data set at hand and regardless of the model being used [8]. When a given method shows a small training MSE but a large test MSE, it is the case of overfitting which is discussed later. However, no matter if overfitting exists or not, the training MSE is always expected to be smaller than the test MSE.

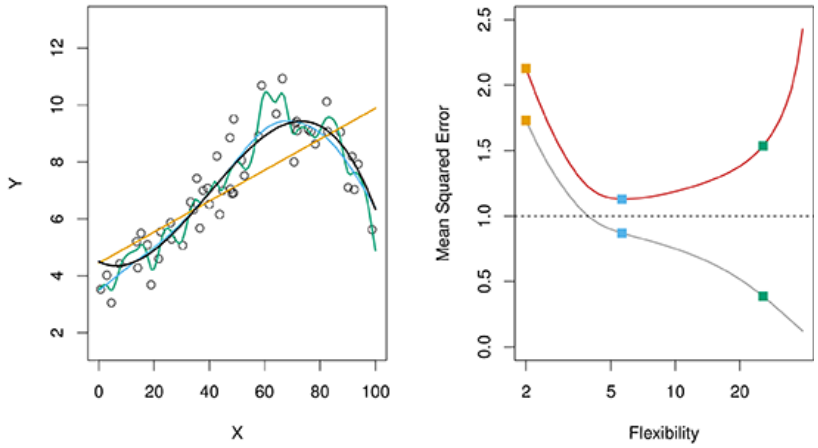

Figure 1: Left: Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves). Right: Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel. [8]

In practice, it is easier to compute the training MSE than the test one since we do not have test data. In this report, a number of methods will be introduced to minimize this quantity.

Another parameter can be used to measure the fit of the model which is called R^2 computed by r2score function. This represents the percent of variance in our dependent variable that can be

explained by our model. To be more explicitly,

$$R^2(\boldsymbol{y}, \hat{\boldsymbol{y}}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y})^2} \tag{3}$$

where $\bar{y}$ is the mean value of $\boldsymbol{y}$ defined as

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i \tag{4}$$

Ideally, we would like to have an R^2 equals 1. If this quantity is zero, our fitted regression line would be a baseline model meaning that it predicts every value of y will be the mean value of y. Occasionally, it shows a negative result revealing the worst model.

R-Squared and MSE are the fundamental statistics to use when determining if our regression model can accurately predict a variable. There are more other statistics like mean absolute error or root mean squared error that can be used to measure our prediction error, in this study, however, we concern about the two mentioned metrics, MSE and R^2.

2.2 Scikit learn's package

Scikit-learn is a fast and comprehensive package which provides a range of supervised and unsupervised learning algorithms. Its application can be in classification, linear regression analysis, clustering and dimensionality reduction via a consistence interface in Python. Also, it is easy and intuitive to use since it contains the good metric support to allow us to assess the estimate performance. For resampling techniques like bootstrap, we do not have to implement this method manually but the bootsrap samples of the dataset can be automatically created thanks to this useful library. Furthermore, ScikitLearn has an own function for splitting the data in a training set and a test set and also for extracting the features from data to define the attributes in image or text data.

Interestingly, Scikit-Learn has several functions which allow us to re-scale the data. Since input variables may have different units resulting in differences in the scales, this may arise the difficulties of the problem being modeled. Therefore, scaling is a critical step in helping to normalise the data within a particular range and also speeding up the calculations in an algorithm sometimes. The

StandardScaler function in Scikit-Learn ensures that scaled data has zero mean and unit variance. This scaling however has a disadvantage that it does not guarantee that we have a particular maximum or minimum in our data set. That is why another alternative standardization which shrinks the range between zero and one, MinMaxScaler, is introduced. Within a dataset, there will be some data points which are extremely different from the rest which are referred as outliers. If our data contains many outliers, scaling using mean and variance like the StandardScaler function will not work effectively. In this case, RobustScaler using the median and quartiles can be a better way to drop these outliers. Different from StandardScale which works on feature columns, the Normalizer is applied to each sample, i.e., row. In particular, it scales each value by dividing each value by its magnitude in n-dimensional space for n number of features. All data points are now formed in a circle or sphere whose radius is 1 [4]. This normalization is mostly useful for controlling the size of a vector in an iterative process, e.g. a parameter vector during training, to avoid numerical instabilities due to large values.

2.3 Ordinary Least Square

Linear Regression is a very popular approach for supervised learning. In particular, supervised machine learning tasks are basically divided into classification and regression, and Linear Regression belongs to the latter group. Compare with classification where the response variable is qualitative (like 'yes or no', 'healthy or unhealthy', etc), regression involves a numerical and continuous target. For example, in our main problem where we desire to predict the price of the houses, the output of our model is supposed to be quantitative and continuous.

Regression tasks can be separated into two different categories: Simple Linear Regression and Multiple Linear Regression. The first one is a very straightforward method for predicting a quantitative outcome y based on a single predictor variable x whereas the latter group depends on more than one independent variable. We will analyze the case of various predictor variable because of its popularity.

Before we proceed, let remind about the main point in regression is that we want to build a model showing the prediction for the unknown dependent variable y based on the known independent variables x. To be specific, we desire to fit a set of data $\boldsymbol{y} = [y_0, y_1, ..., y_{n-1}]$ from a series of variables

$\boldsymbol{x} = [x_0, x_1, ..., x_{n-1}]$ i.e $y_i = y(x_i)$ when $i = 0, 1, 2, ..., n-1$. For that, a prediction for the values of y which are not in the initial data set will be derived. The most simple way to achieve this is building a fitting function $f(x_i)$ or $\hat{y}$ such that

$$y_i = f(x_i) + \epsilon_i = \hat{y}_i + \epsilon_i = \sum_{j=0}^{p-1} \beta_j x_i^j + \epsilon_i, \tag{5}$$

where x_i^j represents the jth predictor in the ith case and β_j is a regression parameter which quantifies the relation between that predictor variable and the outcome and ϵ_i is a normally distributed noise expressed as $N(0, \sigma^2)$ in which mean value $\mu = 0$ and variance is σ^2. This error is necessarily added since the true relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ may not be linear, there might be other factors affecting y and being independent of x.

The above assumption is for the case of p independent predictors of n cases, for simplicity, we will set n=p. The prediction function $\hat{y}$ can be expressed as follows

$$\hat{y_0} = \beta_0 + \beta_1 x_0^1 + \beta_2 x_0^2 + ... + \beta_{n-1} x_0^{n-1} \tag{6}$$

$$\hat{y_1} = \beta_0 + \beta_1 x_1^1 + \beta_2 x_1^2 + ... + \beta_{n-1} x_1^{n-1} \tag{7}$$

$$\hat{y_2} = \beta_0 + \beta_1 x_2^1 + \beta_2 x_2^2 + ... + \beta_{n-1} x_2^{n-1} \tag{8}$$

$$..... \tag{9}$$

$$\hat{y}_{n-1} = \beta_0 + \beta_1 x_{n-1}^1 + \beta_2 x_{n-1}^2 + ... + \beta_{n-1} x_{n-1}^{n-1} \tag{10}$$

In a vector form,

$$\hat{\boldsymbol{y}} = [\hat{y_0}, \hat{y_1}, \hat{y_2}, ..., \hat{y}_{n-1}]^T, \tag{11}$$

$$\boldsymbol{y} = [y_0, y_1, y_2, ..., y_{n-1}]^T, \tag{12}$$

and

$$\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, ..., \beta_{n-1}]^T, \tag{13}$$

and

$$\boldsymbol{\epsilon} = [\epsilon_0, \epsilon_1, \epsilon_2, ..., \epsilon_{n-1}]^T, \tag{14}$$

and the design matrix $\boldsymbol{X}$ which contains a number of predictor variables p is

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_0^1 & x_0^2 & \cdots & \cdots & x_0^{n-1} \\ 1 & x_1^1 & x_1^2 & \cdots & \cdots & x_1^{n-1} \\ 1 & x_2^1 & x_2^2 & \cdots & \cdots & x_2^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n-1}^1 & x_{n-1}^2 & \cdots & \cdots & x_{n-1}^{n-1} \end{bmatrix} \tag{15}$$

The above design matrix is also called a Vandermonde matrix. From this matrix, equation 5 can be defined as

$$\hat{\boldsymbol{y}} \quad = \quad \boldsymbol{X}\boldsymbol{\beta} \tag{16}$$

or

$$\boldsymbol{y} = \hat{\boldsymbol{y}} + \boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{17}$$

In practice, the error vector $\boldsymbol{\epsilon}$ and the regression parameter vector $\boldsymbol{\beta}$ in this equation are unknown. So if we want to make predictions, we must use data set to estimate the coefficients first. The main point now is to obtain the values of the parameter $\boldsymbol{\beta}$ such that the linear model $\hat{\boldsymbol{y}}$ fits well the real data so that $\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. There are several ways to achieve this, however, by far the most common approach is minimizing the function which involves in the difference between the observed response value y_i and the outcome value predicted by our linear model $\hat{y}_i$. To be more precise, the model will minimize the mean squared error via a so-called Cost function defined as

$$C(\boldsymbol{\beta} = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 = \frac{1}{n}\{(\boldsymbol{y} - \hat{\boldsymbol{y}})^{\boldsymbol{T}}(\boldsymbol{y} - \hat{\boldsymbol{y}})\}, \tag{18}$$

From Eq.16, we can rewrite the above equation as

$$C(\boldsymbol{\beta}) = \frac{1}{n}\{(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})\}. \tag{19}$$

In order to find the optimal parameter β_j, we need to minimize the cost function $C(\boldsymbol{\beta})$, that is

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}\{(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\}. \tag{20}$$

The expression of powers of x in set of equations 6-10 can be replaced with elements of Fourier series or $\cos(jx_i)$ or $\sin(jx_i)$, or time series or other orthogonal functions. Generally, we can interpret Eq.(20) as

$$\frac{\partial C(\boldsymbol{\beta})}{\beta_j} = \frac{\partial}{\partial\beta_j}\left[\frac{1}{n}\sum_{i=0}^{n-1}(y_i-\beta_0 x_{i,0}-\beta_1 x_{i,1}-\beta_2 x_{i,2}-\cdots-\beta_{n-1}x_{i,n-1})^2\right]=0, \tag{21}$$

which leads to

$$\frac{\partial C(\boldsymbol{\beta})}{\beta_j} = -\frac{2}{n}\left[\sum_{i=0}^{n-1} x_{ij}(y_i-\beta_0 x_{i,0}-\beta_1 x_{i,1}-\beta_2 x_{i,2}-\cdots-\beta_{n-1}x_{i,n-1})\right]=0, \tag{22}$$

or to a matrix-vector form

$$\frac{\partial C(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} = 0 = \boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}). \tag{23}$$

If the matrix $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, we can derive the optimal values of parameter $\boldsymbol{\beta}$ as

$$\boldsymbol{\beta}^{opt} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{24}$$

so that $\hat{y} = \boldsymbol{X}\boldsymbol{\beta}^{opt}$. The procedure of finding these parameters to minimize the mean square error is known as Ordinary Least Square.

2.4 Rigde regression

So far, OLS method choose the coefficients for the best fit of data. However, it gives us the unbiased coefficients which means it treats all the independent variables as the same importance. In other words, its aim is just simply to obtain the set of parameters $\boldsymbol{\beta}$ as long as they reduces the mean square error at least. This causes an overfitting problem (Fig. 2) in which the model fits the training data too well and thus leads to a negative impact on modelling the test data. For this reason, we have to find out a biased model which consider its variables unequally to make a better

generalization on both training and test data. In addition, another typical linear regression problem so-called singularity can be arisen when design matrix $\boldsymbol{X}$ is high-dimensional (the number of features exceeds the number of cases). That is, in particular, when $\boldsymbol{X}$ has linearly dependent column vectors, referred as super-collinearity, it is impossible to find out the inverse of $\boldsymbol{X}^T\boldsymbol{X}$, leading to the unsolved parameter β_j. This problem can be addressed if we add a small diagonal component to the matrix which involves a hyperparameter called λ. This is when the Ridge regression is introduced.

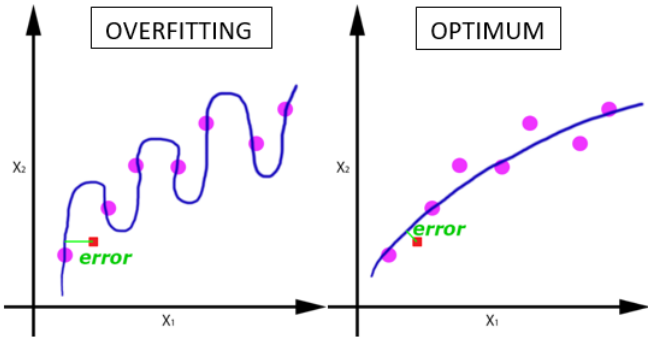

Figure 2: Overfitting vs Optimal case [7].

Recall from section 2.3, the OLS aprroach is interpreted as

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}\{(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\}, \tag{25}$$

and it can be simply seen that

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \frac{1}{n}\sum_{i=0}^{n-1}(y_i-\hat{y}_i)^2 = \frac{1}{n}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2, \tag{26}$$

by using a norm-2 vector which has the following definition

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_i x_i^2}. \tag{27}$$

In the end, we will obtain the analytical expression for the fitting parameter $\boldsymbol{\beta}$ by minimizing the above equation. That is basically what OLS does. Compared with OLS, Ridge regression is very similar, except that the coefficients are estimated by minimizing a slightly different quantity.

Particularly, the Ridge regression parameter $\boldsymbol{\beta}^R$ are obtained by minimizing the new cost function as

$$\frac{1}{n}\|\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2+\lambda\|\boldsymbol{\beta}\|_2^2, \tag{28}$$

where $\lambda \geq 0$ is a regularization or a tuning parameter. Ridge regression has the same purpose like OLS, that is to choose the coefficient estimates that fit the data well by making the MSE as small as possible. However, the term with lamda called a shrinkage penalty is small when $\beta_1, \cdots, \beta_{n-1}$ are close to zero and so it has the effect of shrinking the estimates of β_j towards zero.Therefore, the regularization parameter λ is added to the least square cost function to monitor the relative impact of these two terms in the above equation on estimating the regression coefficients. In order to find out the values for $\boldsymbol{\beta}$, we would like to express the cost function in matrix-vector form,

$$C(\boldsymbol{X},\boldsymbol{\beta})=\{(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})\}+\lambda\boldsymbol{\beta}^T\boldsymbol{\beta}, \tag{29}$$

we then let the derivative of $C(\boldsymbol{X},\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ be zero

$$\begin{aligned}\frac{\partial C(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} &= 0=2\boldsymbol{X}^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\beta})-2\lambda\boldsymbol{\beta} &(30)\\ \Rightarrow \boldsymbol{\beta}^{\boldsymbol{R}} &= (\boldsymbol{X}^T\boldsymbol{X}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}, &(31)\end{aligned}$$

where $\boldsymbol{I}$ is a $p\times p$ identity matrix with the constraint that

$$\sum_{i=0}^{p-1}\beta_i^2\leq t, \tag{32}$$

with t a finite positive number. In fact, when $\boldsymbol{X}$ is orthogonal, that is

$$\boldsymbol{X}^T\boldsymbol{X}=(\boldsymbol{X}^T\boldsymbol{X})^{-1}=\boldsymbol{I}. \tag{33}$$

We can obtain the relation of parameter vector β from two methods

$$\boldsymbol{\beta}^{Ridge}=(\boldsymbol{I}+\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}=(1+\lambda)^{-1}\boldsymbol{\beta}^{OLS} \tag{34}$$

With the expression in Eq.34, we can clearly see the key role of λ in controlling the magnitude of the regression parameters. When $\lambda = 0$ (no penalty), the rigde $\boldsymbol{\beta}$ becomes the same as OLS $\boldsymbol{\beta}$. However, as $\lambda \to \infty$, the impact of skrinkage penalty grows, and the Ridge regression parameters will approach zero, but they can not be zero. That is because Ridge regression puts different importance to the features but does not ignore the unimportant ones. There is another interesting point in Ridge regression is that it generates not only one set of fitting parameters but several depending on the value of λ. Selecting a good value for λ to minimize MSE is thus vital to achieve the best model.

2.5 Bias-variance trade-off

We see that Ridge regression is just another version of the OLS with an additional diagonal term in the design matrix $\boldsymbol{X}$. The interesting point lies on its consequences, bias-variance trade-off.

Consider a dataset $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ including n pairs of independent and dependent variables. Rewrite the equation 5 to show the relation between true data $\boldsymbol{y}$ and fitting model $f(\boldsymbol{x})$, that is

$$\boldsymbol{y} = f(\boldsymbol{x}) + \boldsymbol{\epsilon} \tag{35}$$

In OLS regression, we define the fitting function $\hat{y} = \boldsymbol{X}\boldsymbol{\beta}$ and the estimator $\boldsymbol{\beta}$ is chosen by minimizing a cost function which related to MSE

$$C(\boldsymbol{X}, \boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 = \mathbb{E}[(\boldsymbol{y} - \hat{\boldsymbol{y}})^2], \tag{36}$$

by using the definition of expected value $\mathbb{E}[\boldsymbol{x}] = \frac{1}{n}\sum_i x_i$.

We can write the above equation as

$$\mathbb{E}[(\boldsymbol{y} - \hat{\boldsymbol{y}})^2] = \frac{1}{n}\sum_i (f_i - \mathbb{E}[\hat{\boldsymbol{y}}])^2 + \frac{1}{n}\sum_i (\hat{y}_i - \mathbb{E}[\hat{\boldsymbol{y}}])^2 + \sigma^2 \tag{37}$$

To derive this equation, we need to consider the variance of $\boldsymbol{y}$ equals the variance of $\boldsymbol{\epsilon}$, that is σ^2. Also, the mean value of ϵ is zero and the function f is equivalent to $\hat{y}$, we will obtain a more compact

notation for the expected value

$$\mathbb{E}[(\boldsymbol{y} - \hat{\boldsymbol{y}})^2] = \mathbb{E}[(\boldsymbol{f} + \boldsymbol{\epsilon} - \hat{\boldsymbol{y}})^2]. \tag{38}$$

By adding and subtracting $\mathbb{E}[\hat{\boldsymbol{y}}]$, we get

$$\mathbb{E}[(\boldsymbol{y} - \hat{\boldsymbol{y}})^2] = \mathbb{E}[(\boldsymbol{f} + \boldsymbol{\epsilon} - \hat{\boldsymbol{y}} + \mathbb{E}[\hat{\boldsymbol{y}}] - \mathbb{E}[\hat{\boldsymbol{y}}])^2], \tag{39}$$

The above expected value then can be rewritten as

$$\mathbb{E}[(\boldsymbol{y} - \hat{\boldsymbol{y}})^2] = \mathbb{E}[(\boldsymbol{y} - \mathbb{E}[\hat{\boldsymbol{y}}])^2] + \mathbb{E}[(\hat{y} - \mathbb{E}[\hat{\boldsymbol{y}}])^2] + \sigma^2. \tag{40}$$

We can interpret all the terms in the expected test MSE respectively as

$$\mathbb{E}[(\boldsymbol{y} - \hat{\boldsymbol{y}})^2] = Bias^2 + Var + Var(noise), \tag{41}$$

since $Var(noise) = \sigma^2$. This equation tells us that if we want to minimize the expected test error, a method where low variance and low bias are obtained is needed. Note that since variance and squared bias are always nonnegative, the expected test MSE will never be lower than variance σ^2 of the noise. The question now is what variance and bias really mean? Variance refers to the amount by which $\hat{\boldsymbol{y}}$ would change when we estimate it using a universe of different training data. That means different set of training data will lead to different $\hat{\boldsymbol{y}}$. The ideal case is when the estimate for $\hat{\boldsymbol{y}}$ does not fluctuate too much among training sets. High variance is understood as a small change in training data can change significantly $\hat{\boldsymbol{y}}$. On the other hand, bias measures the deviation of the expectation value of our estimator $\hat{\boldsymbol{y}}$ from the true value $\boldsymbol{y}$. In short, it can be said that variance is related to a model failing to fit the test set while bias involves in a model failing to fit the training one. These two quantities show a trade-off relationship over the model complexity, particularly, as we use a more complex model, the variance will increase while the bias will be in opposite trend (Fig.3). Thus, optimal performance is achieved at intermediate levels of model complexity. This is when Ridge regression improves over OLS. It can be observed from the Fig.3 that OLS model is always

on the rightest of the figure since its estimation is unbiased meaning that the more independent variables the model possesses, the more complex it is. It thus always gives the lowest bias and the highest variance which is not what we desire to. However, this issue can be fixed by Ridge regression as we can tune the λ parameter so that model coefficients $\boldsymbol{\beta}$ vary. In particular, as λ increases, the

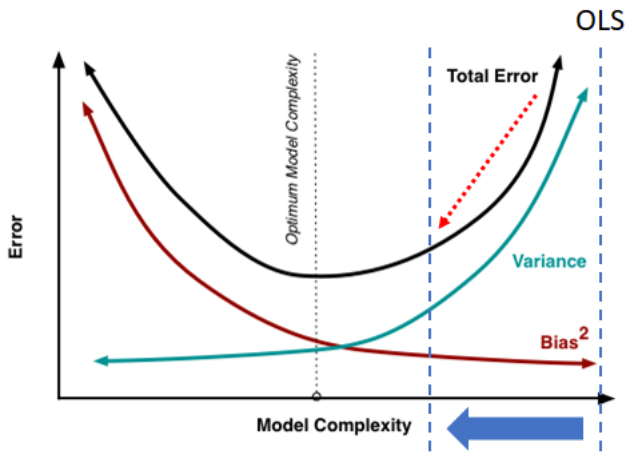

Figure 3: Bias-Variance trade-off and model complexity [5].

Ridge regression model becomes simpler, resulting in decreased variance but increased bias. The case in Fig.4 using a data set consisting of 45 features in 50 cases. At $\lambda = 0$ or at OLS regression, we obtain a high variance and no bias. But when λ increases, the shrinkage of the Ridge coefficient estimates reduces steadily the variance of the predictions while bias is enhanced. Take a look at the test MSE line which is basically a function of sum of the variance and squared bias, we can observe its notable decrease as λ increases from 0 to 10. This is due to the fact that in this range of λ, the variance drops rapidly whereas the bias witnesses a small increase. As λ continues to grow, the decrease in variance is observed and the shrinkage on the coefficients underestimate them leading to an escalation in the bias. The minimum MSE is achieved at approximately $\lambda = 30$. A remarkable point in this figure is that at $\lambda = 0$ and $\lambda = \infty$ corresponding to the OLS model and null model where all estimators are zero, we acquire the similar MSE. In the end, MSE is considerably lower at an intermediate value of λ. In general, in the case that the number of features p approximately equals the number of samples n, the OLS estimates can be variable as in this example. Besides, when n falls behind p, i.e, $p > n$, OLS estimates do not even have a unique solution while Ridge

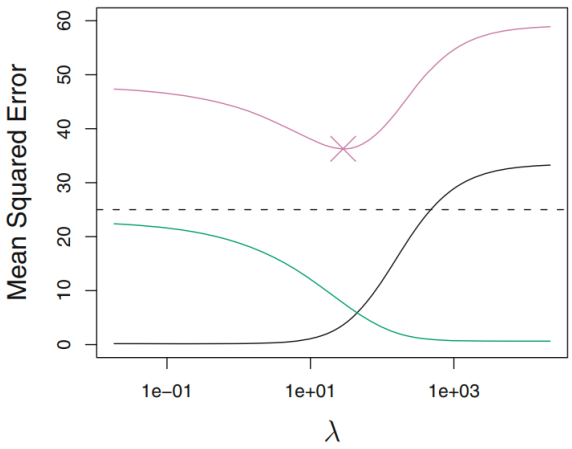

Figure 4: Squared bias (black), variance(green), and test mean squared error(purple) for the Ridge regression predictions on the simulated data set, as a function of λ. The horizontal dashed line indicate the minimum possible MSE. The purple crosses indicate the Ridge regression models for which the MSE is smallest [8].

regression can still show a good performance thanks to the bias-variance trade-off. In short, Ridge regression would shine in situations where OLS model obtain high variance.

In summary, the bias-variance trade-off is a central problem in supervised learning in which we want to choose a ideal model capturing the regularities in the training data and simultaneously generalizing well to test data. In fact, regularization method like Ridge regression can somehow reduce variance considerably compared to OLS estimates leading to a better MSE performance.

2.6 Resampling techniques

In previous section, we mention about splitting our data into training and testing sets. In fact, the test set is not used to assess model performance during the training phase. So how can we generalize our model?

- Option 1: assess an error metric based on the training data. Unfortunately, this sometimes can make the model overfitting which performs very well in training set but fail to fit the test one (Fig.2).

- Option 2: validation approach. In this case, we separate the training set into two groups: a training set and a validation set. We thus can fit the model on the new training data and the

fitted model is used to predict the responses for the observation in the validation set. The resulting validation set error rate provides an estimate of the test error rate. The disadvantage of this method is that we use just a single validation set which is not reliable if we are working with a small size of data. In practice, there is a limit in data set so this problem matters.

- Option 3: (also an optimal option) resampling methods. This involves drawing repeatedly samples from a training set and refitting a model of interest on each sample. This approach gives us the additional information about the fitted model which can not be derived if we train our data only once. The two most widely used resampling methods are cross-validation and bootstrap.

2.6.1 Cross Validation

Recall the validation approach, only one subset of the samples is used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this means the validation set error rate may overestimate the test error rate for the model fit on the whole data. For this, cross validation is proposed as a refinement of the validation method.

2.6.2 Leave-One-Out Cross Validation

Leave-One-Out Cross Validation(LOOCV) is based on validation approach which also divides the set of samples into two parts. The only different point is that two subsets will no longer have a similar size but the validation set contains only one sample (x_1, y_1) and the rest of samples $\{(x_2, y_2), ..., (x_n, y_n)\}$ belong to the training set. The procedure of selecting validation data and training set will be continued with (x_2, y_2) and $\{(x_1, y_1), (x_3, y_3), ..., (x_n, y_n)\}$ respectively. This is illustrated in Fig. 5. For the i^{th} of selecting, we acquire a test MSE, that is $MSE_i = (y_i - \hat{y}_i)^2$. Carry out this procedure n times, n test MSE will be obtained. The LOOCV estimate for the test MSE is the average of these n test MSE estimates:

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} MSE_i \tag{42}$$

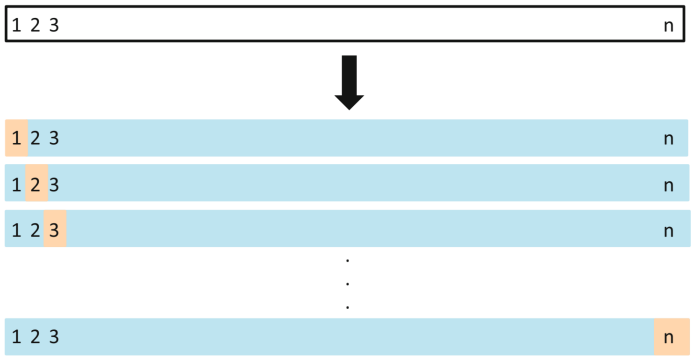

Figure 5: A schematic display of LOOCV. A set of n data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the n resulting MSE's. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth [8].

In short, LOOCV can fix the problem of validation approach. First, since in LOOCV, we repeatedly fit the model using training sets that contain $n-1$ samples up to n times, this approach tends not to overestimate the test MSE like validation method does. Second, in LOOCV, there is no randomness in the training/validation set splits as we perform this several times which is in contrast to validation method.

2.6.3 K-folds Cross Validation

An alternative to LOOCV is k-fold Cross Validation (k-fold CV) which involves randomly dividing the training data into k groups (or folds) of approximately equal size. The first fold, is treated as validation set, is used to compute model performance and the model is fit on the remaining $k-1$ folds. This procedure is repeated k times; each time, a different fold is treated as the validation set. Similarly as LOOCV, for k times of repeating, k estimates of the test MSE, that is $MSE_1, MSE_2, ..., MSE_k$. The k-fold CV estimate is then computed by averaging the k test MSE, providing us with an approximation of the error we might expect on unseen data,

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i. \tag{43}$$

Consequently, with k-fold CV, every set of training data will become validation one once which is illustrated in Fig.6. In practice, typical choice for k is 5-10. Actually, there is no formal rule about

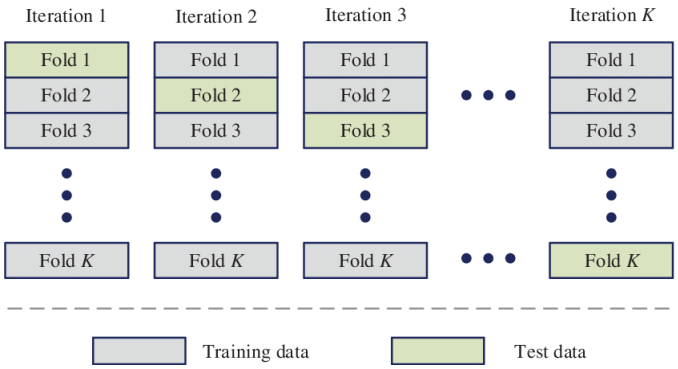

Figure 6: Illustration of the k-fold CV process [6].

the size of k; however, if k gets bigger, the difference between the estimated performance and the true performance to be seen in the validation set will decrease. Also, we will pick $k = 5$ or 10 rather than $k = n$ (LOOCV) is because of the computational burdens.In particular, LOOCV requires fitting the model in n times which is computationally expensive. In contrast, performing 5 or 10-fold CV requires fitting the learning method only 5-10 times which may be much more feasible.

Make a comparision with validation approach,in Fig.7, we can see the variability in test MSE from 10-fold CV is much lower than the variability resulting from validation method. It is thus said that k-fold CV can help to enhance the accuracy of the estimated MSE. In other words, for the small size of data ($n < 10000$), 5 or 10-fold CV can improve the precision of your estimated performance.

2.6.4 Bootstrap

Although using $k = 5 - 10$ helps to minimize the variability in the estimated performance, k-fold CV still tends to have higher variability than bootstrapping which is another common resampling technique introduced in this section. A bootstrap sample is a random sample of the data taken with replacement [3]. This means that, if a data point is selected for inclusion in the subset, the probability that it will be picked in later selection is nonzero. Now, we will briefly reveal how it works to provide a measure of accuracy of the parameter estimate or of a given statistical learning method through some steps as follows:

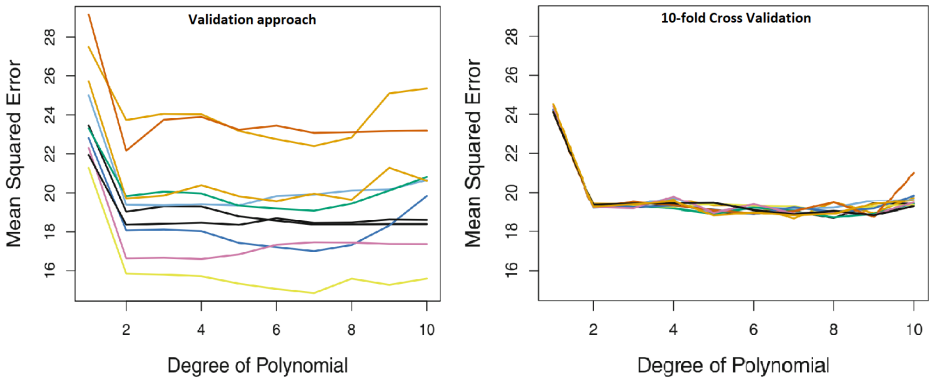

Figure 7: Validation approach and k-fold CV are used on the specific data set in order to estimate the test MSE using polynomial functions.Left: The validation method is repeated 10 times, each time using a different random split of the samples into a training set and validation set. This illustrates the variability in the test MSE that results from this approach. Right: 10-fold CV is run 9 separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves [8].

1. Draw with replacement n numbers for the observed variables

   $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$

2. Define a bootstrap sample $\boldsymbol{x}^*$ containing the values which are drawn from original data set $\boldsymbol{x}$.

   For example, $\boldsymbol{x}^*$ might be $\boldsymbol{x}^* = (x_3, x_1, x_1, x_{10}, ..., x_{n-1}, x_{n-1})$.

   Note that $\boldsymbol{x}^*$ possibly contains repetitive element and the size of $\boldsymbol{x}$ and $\boldsymbol{x}^*$ are equal, thus, some variables can be picked for bootstrap sample several times whereas some are not chosen even once.

3. Rely on the vector $\boldsymbol{x}^*$, the expected value ϵ_1^* is computed.

4. Repeat this process as many as we want, for instance, k times.

5. Finally, using the k bootstrap sets to get the bootstrap estimate of the quantity of interest. In this case, it is the final expected value,

$$\boldsymbol{\epsilon}^* = \frac{1}{k} \sum_{i=1}^{k} \epsilon_i^*. \tag{44}$$

Figure 8 provides an example of 12 variables in bootstrap sample which can been seen that not only

the size but the distribution of values in bootstrap sample (shown in colors) is similar as in original set. It is estimated that about 63.42% of the original sample ends up in any particular bootstrap sample. The variables from $\boldsymbol{x}$ which are not included in a specific bootstrap sample are referred as the "out-of-bag" samples. They are used for assessing the model while the selected ones will build the model. As we mentioned at the beginning of this section, there is a lower variability in test MSE

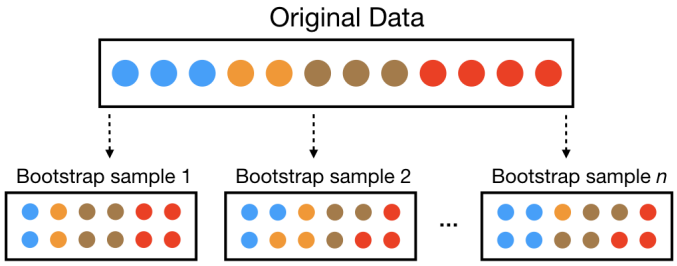

Figure 8: Illustration of the bootstrapping process [2].

in bootstrap than in k-fold CV. This is due to the fact that the observed variables are replicated in bootstrapping. However, this means that a higher bias will be present in your error estimate which could be a serious issue if the size of data set is small. For the data set is medium or large ($n \geq 1000$), this problem can be ignored.

Bootstrap is a widely applicable and extremely powerful statistical tool thanks to its simplicity. It can be easily applied to a variety of statistical learning methods to derive estimates of standard errors and confidence intervals for complex estimators of the distribution. The power of bootstrap also lies on giving a more accurate inferences even when the data set is not well behaved or when its size is small since the approach does not depend on distributional assumptions.

3 Code, implementation and testing

3.1 OLS on Franke function

The central objectives of this part is to study Ordinary Least Squares (OLS) method which discussed in the previous section. First, we need to define Franke's function which has two Gaussian peaks of different heights ans a smaller dip. This function is widely used for testing the interpolation and

fitting algorithms. Figure 9 describes the Franke function formed by the expression as follows

$$\begin{aligned} f(x) = 0.75\exp\left(-\frac{(9x_1-2)^2}{4} - \frac{(9x_2-2)^2}{4}\right) + 0.75\exp\left(-\frac{(9x_1+1)^2}{49} - \frac{9x_2+1}{10}\right) \\ +0.5\exp\left(-\frac{(9x_1-7)^2}{4} - \frac{(9x_2-3)^2}{4}\right) - 0.2\exp\left(-(9x_1-4)^2 - (9x_2-7)^2\right) \end{aligned} \tag{45}$$

We now want to apply OLS without any resampling techniques in our data. For that, a random data

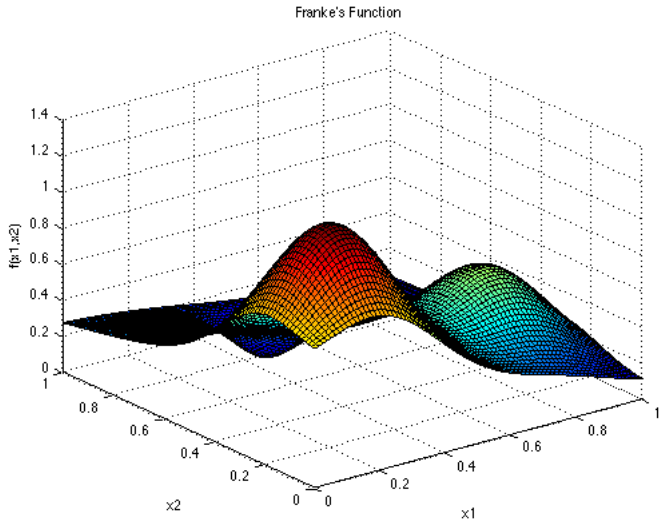

Figure 9: Franke function .

$x, y \in [0, 1]$ will be formed by Franke function combining with the stochastic noise. Figure 10 illustrates our original data in three dimensions. Next, a matrix X will be formed which has $20 \times 20 = 400$ samples and 2 features(x and y). Since we desire to try out a polynomial fit with x and y dependence of the form $[x, y, x^2, y^2, xy, ...]$, *PolynomialFeatures* function from *sklearn.preprocessing* package is used for our design matrix. Spitting dataset into training and test set and scaling dataset is the next essential steps. Again, we take advantage of Scikit learn's package like *sklearn.model_selection* or *sklearn.preprocessing* to do these steps. Throughout the whole problem, we will use StandardScaler as a fundamental scaling tool. Now, the OLS linear regression which is from Scikit learn will be applied to fit the scaled design matrix for training set with our original training data. Following that, we train the model in both training and test data set. Last but not least, the model will be

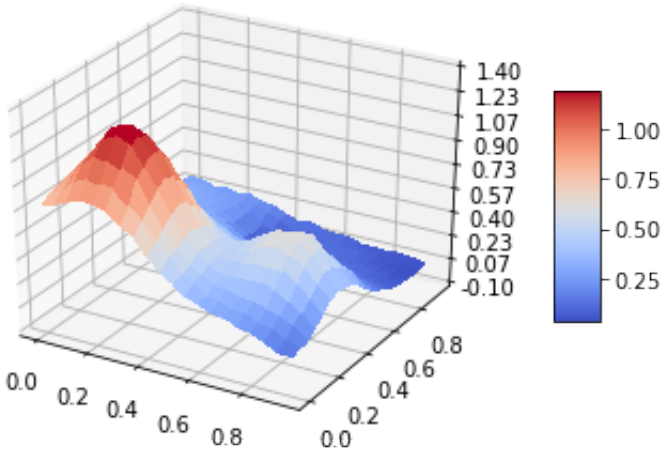

Figure 10: Franke function disturbed by stochastic noise .

evaluated by measuring the training MSE, test MSE and R^2 score which automatically derived from Scikit learn's library. Note that it might be useful to vary the polynomial degree (for example, from 1 to 15) so that we can explore which polynomial fits the data best.

3.2 OLS and Bootstrap

Bootstrap is the resampling method that will be used first to study the bias-variance trade-off in the context of continuous predictions like OLS regression. 100 bootstrap samples are created to derive the test MSE, bias and variance of the model when the polynomial degree increases one by one up to 15.

3.3 OLS and k-fold Cross Validation

We apply k-fold cross validation for this part where k is set at 5-10 resulting a split in 5-10 sets for training and test data. Following this, it is essential to scale the data in order to avoid data leakage problem. Based on Eq.43, we derive the the test MSE in 5-10 estimates by using KFold from *sklearn.model_selection*. In the end, we make a comparision about OLS test MSE betweem two resampling techniques: Bootstrap and k-fold CV.

3.4 Ridge regression on the Franke function with resampling techniques

First, instead of using OLS, we now apply Ridge regression on Franke function. Once again, this can be easily performed with package *sklearn.linear_model*. We set the value of hyperparameter $\lambda = 0.001$ on this method and make a comparision with OLS approach at a specific degree, for instance, at 5. Secondly, we perform the bootstrapping with the same polynomial degrees to infer the bias-variance trade-off as in OLS case. Furthermore, with a group of 100 tuning parameters λ which ranges from 10^{-3} to 10^{5}, we use 100 bootstrap samples for each λ to achieve the test MSE, bias and variance and study the bias-variance trade-off as a function of various values of tuning parameter λ. Finally, we combine Ridge with k-fold CV in different values of λ to define which value of this hyperparemeter gives us the minimal test MSE. For that, two approaches from sklearn which have KFold and *cross_val_score* along with KFold are used.

3.5 Boston Housing data set

We now apply the codes done on Franke function to the real data from Boston Housing data set. The dataset can be imported from scikit-learn which contains 13 features and 1 target called $'MEDV'$. We can check the dataset description using $boston\_data.DESCR$. The aim for this exercise is to perform polynomial regression using both methods mentioned above, OLS and Ridge to find the best model for predicting the value of $MEDV$. For that, all the metrics for assessing the quality of a model, MSE and R^2 are used. A good start is to look at the linear correlation between features illustrated in a correlation matrix by importing *seaborn* package. More specific, *seaborn* is a robust library which provides a high-level interface for drawing attractive and informative statistical graphics. After that, a function to get correlated feature data with a specific threshold will be made so that we can select which are the features having high correlation with the target. In our project, we let threshold be 0.5 and scatter plots for selected features are then shown.

Before we select relevant features, a model using all the features is referred as a baseline. Firstly, we perform OLS Regression on the matrix formed by all the features by using scikit-learn package and then verify the metrics if they are good. This first step is just to try with the original matrix, the more interesting thing lies on using scikit-learn's functionality, $PolynomialFeatures$, to create

the design matrix and apply OLS on it. After that, we can combine OLS with CV to check the test MSE and also use different scaler packages (MinMaxScaler and StandardScaler) to find out the most suitable data-scaling methods. Secondly, Ridge regression will be perform with CV and we would see how test MSE will be changed by different value of λ parameter.

Now, we desire to study in our selected features with Ridge. In particular, instead of using all features, we pick some which have high impact on the target. In the end, we get the values of test MSE and R^2. A scatter plot with the true values against the predicted values from our model will be built by using *seaborn* package.

4 Analysis and overall presentation

4.1 OLS on Franke function

As the polynomial degree increases, the training MSE and test one decrease at first, then at degree of 13, we witness an increase in test MSE whereas training MSE still goes down. Depend on this tendency which matches with the theory, we can conclude the model using polynomial degree of 13 can be the optimal model since it obtains the minimal test MSE (Fig. 11).

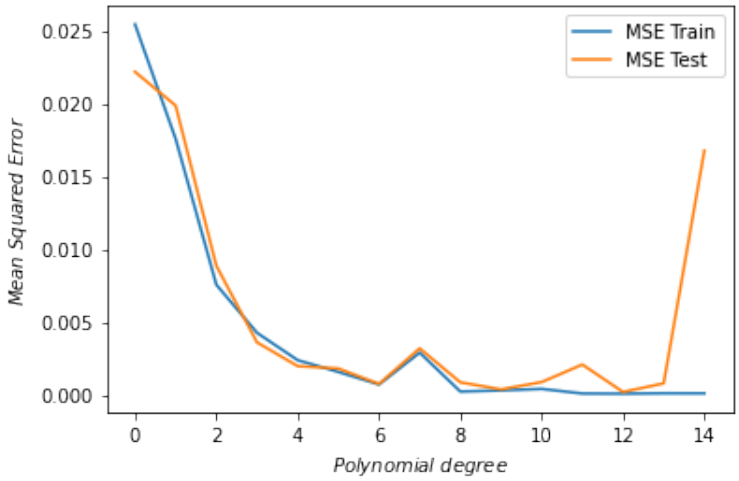

Figure 11: Mean Squared Error as a function of Model Complexity.

4.2 OLS and Bootstrap

We obtain different values of test MSE, bias and variance for different polynomial degrees (Fig.12). As the complexity of the model goes up, the bias falls off because the true function is non-linear while the variance climbs slightly and the bias tends to initially decreases faster than the variance increases. Consequently, the test MSE declines substantially yet have not experienced any significant increase as model flexibility increases. This is indeed the bias-variance trade-off that we introduced in Formalism section.

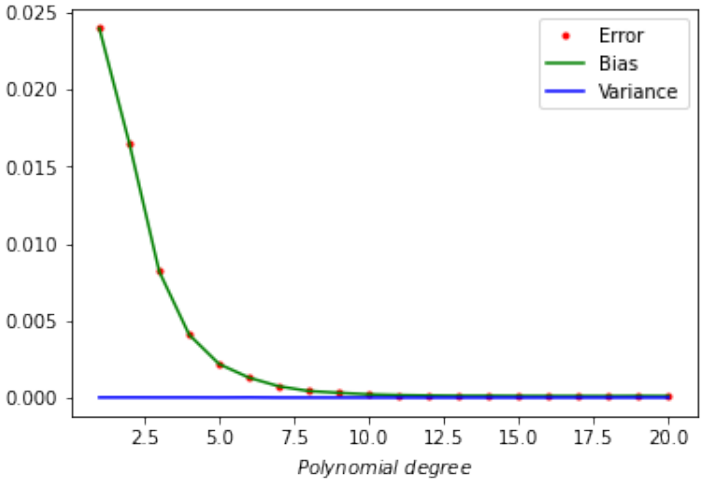

Figure 12: Bias-variance trade-off as a function of model complexity in OLS. Squared bias (green curve), variance (blue curve) and test MSE (red dotline) are derived from our data set with an increase in model complexity.

4.3 OLS and k-fold Cross Validation

Figure 13 indicates different cases of using k (5-10) folds together with an increase of flexibility of the model (up to 20). In these cases, we obtain the MSE resulting from test folds in two different resampling techniques, Bootstrap and k-fold CV. Compared with MSE obtained by bootstrap method, we observe that in CV, there is a more wildly fluctuation giving us a more visible result about which model achieves the smallest test MSE. Furthermore, the MSE from CV shows a higher value which can apparently seen at higher polynomial degree. Unfortunately, in the case $k = 5$ and maximum degree is 15, an extreme amount of test MSE is obtained from CV approach showing an

unreliable result.

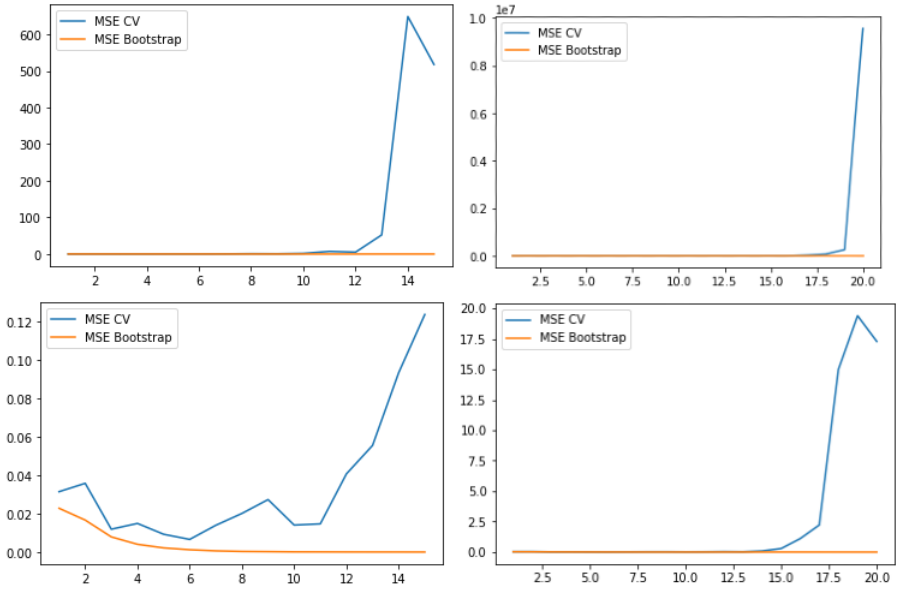

Figure 13: A comparision in test mean squared error in terms of model complexity for 4 different cases in which the k folds and the range of polynomial are varied. Top left: $k = 5$ and $degree = 1-15$; top right: $k = 5$ and $degree = 1 - 20$; bottom left: $k = 10$ and $degree = 1 - 15$; bottom right: $k = 10$ and $degree = 1 - 20$.

4.4 Ridge regression on the Franke function with resampling techniques

Initially, we use Ridge as an alternative regression method. All the metric values for statistics (test MSE, training MSE and R^2 score) are obtained in the case the polynomial degree equals 5 (Table 1). Compared these results with OLS in the same degree, we see that both methods give us the similar values for test MSE and R^2 score. Furthermore, R^2 approaches to 1 revealing a very good model we are using. However, the training MSE OLS obtained is much higher than Ridge's one.

	OLS	Ridge
test MSE	0.0022930653794029985	0.0026560620556158863
train MSE	0.0023521824768885868	0.0002500132273948464 4
R^2 score	0.9702429182849661	0.9655323148048487

Table 1: Metric values for statistics in OLS and Ridge regreession where polynomial degree is 5 and $\lambda = 0.001$.

As in OLS, we can observe the bias-variance trade-off by using Bootstrapping together with Ridge regression. Indeed, we can perceive from Fig.14 the same behavior of bias and variance like in OLS regression (Fig.12) which makes sense. Also, in Ridge, the bias-variance relationship can be

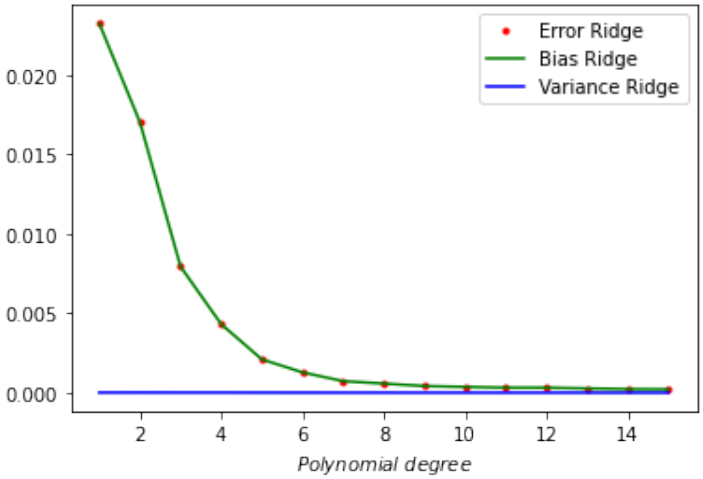

Figure 14: Bias-variance trade-off as a function of model complexity in Ridge regression.

demonstrated in terms of different value of tuning parameter λ. Fig.15 shows that when λ gets bigger meaning the Ridge regression model gets simpler, the squared bias of our ridge estimation on the test data steadily increases along with the test MSE while variance indicates a reversed tendency. When k-fold cross-validation is applied in two different tools (Fig.16), we observe that both of them give us the similar results about minimal test MSE at a particular λ. However, cross-validation using *cross_val_score* from sklearn along with KFold achieves a much smaller values for test MSE.

4.5 Boston Housing data set

Firstly, based on the correlation matrix (Fig.17), LSTAT and RM shows a higher contrast correlation with the MEDV. Another interesting feature is the high negative correlation of the AGE of the houses and the DIS to the Boston employment centers. This may denote that new houses are built farther from the employment centers which makes sense. Besides, DIS also shows a strong negative correlation with NOX concentration and INDUS proportion of non-retail business acres per town. This is an interesting feature assuming that the employment centers are where most of the

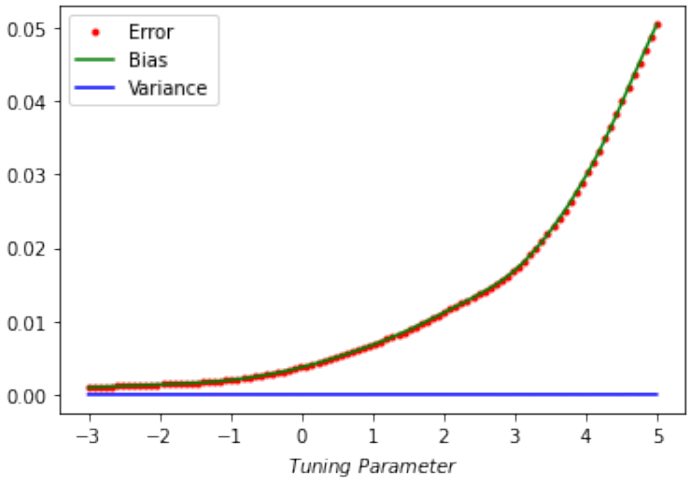

Figure 15: Bias - variance trade-off as a function of tuning parameter λ in Ridge regression .

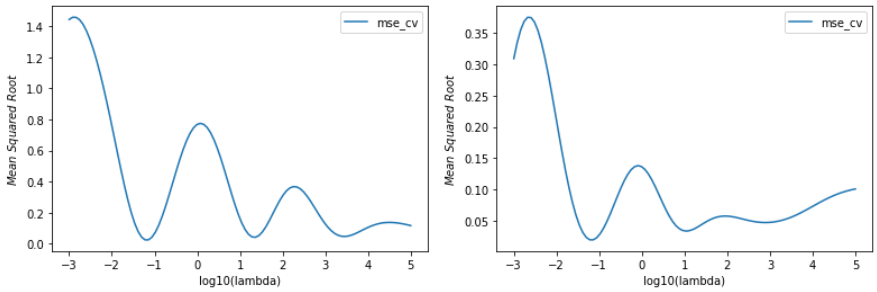

Figure 16: Right to left: Test MSE obtained by cross-validation using only KFold and *cross_val_score* with KFold from sklearn respectively.

concentration of NOX will be and that the farther you are from the employment center, the more non-retail businesses will appear. Plus, TAX feature shows an interesting correlation with RAD, NOX, and INDUS. For this project, if we set the minimum correlation required to be a feature to 0.5, then we will have the following features [RM, LSTAT, PTRATIO] can affect the price of the houses (Table 2). A scatter plot of those features is made (Fig.18). We can also explore the inclusion of the other features such as TAX and DIST and check if our model improves its prediction capability.

When we use a polynomial model of degree 2, this improves the MSE and R2 accuracy of our prediction. However, when the complexity of model increases to degree 3, it yields a very high MSE and R2 scores. We can conclude that degree 2 is the best possible polynomial complexity that we can use.

Figure 17: Correlation matrix for all features.

	Corr Value
RM	0.695
PTRATIO	-0.508
LSTAT	-0.738
MEDV	1.000

Table 2: Correlated features with threshold greater than 0.5

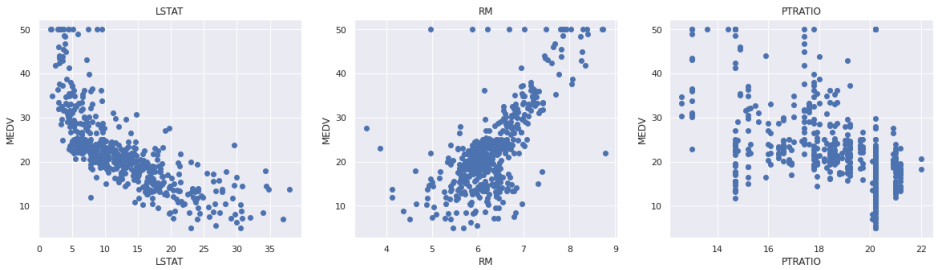

Figure 18: Scatter plot of selected features: LSTAT, RM and PTRATIO respectively.

Different data-scaling methods are applied with OLS shown in Fig.19. As we can see that the StandardScaler in this case is outperformed the MinMaxScaler since it gives us the lower test MSE. In the case of Ridge regression along with CV, the relation between test MSE and λ parameter is studied and demonstrated in Fig.20. From here, we can decide which value of λ is proper for our model.

When we try to remove features that have low correlation with the target and apply Ridge regression on our model, we get a scatter plot showing how good our model is.

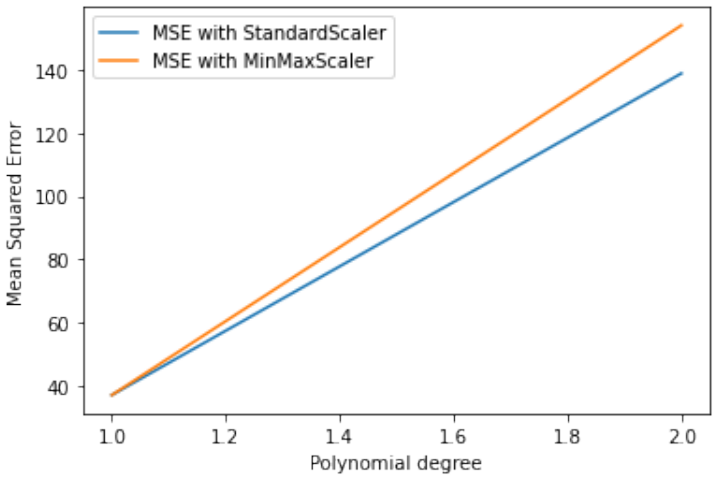

Figure 19: MSE as a function of Model Complexity in 2 different scaling methods: StandardScaler and MinMaxScaler.

5 Conclusions

In conclusion, we have studied in more detail the two common-used regression methods, OLS and Ridge. In the first techniques, an observation about test and training MSE as function of model complexity is obtained. Thanks to these, we achieve the most proper polynomial degree where the smallest MSE presented. By using resampling techniques along with OLS, a more reliable MSE is acquired and the bias-variance trade-off is observed with respect to the model flexibility. Regarding to the two resampling methods, they demonstrate the same results for test MSE but at a more complex model, k-fold CV shows a higher value. The second regression method applied in this

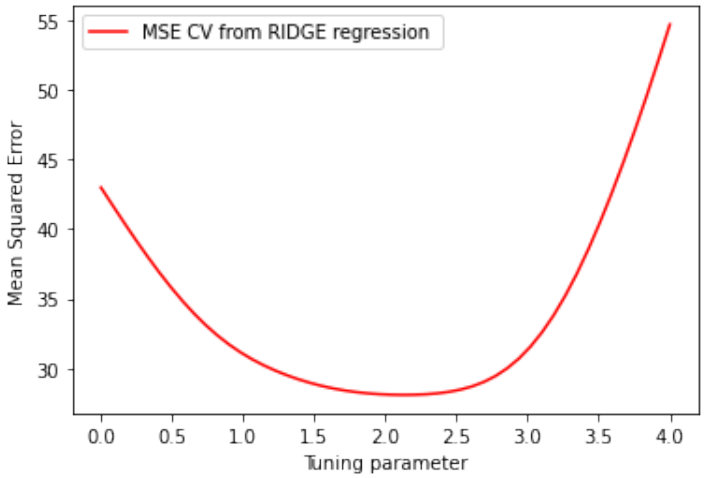

Figure 20: MSE obtained by Ridge regression vary with the tuning parameter.

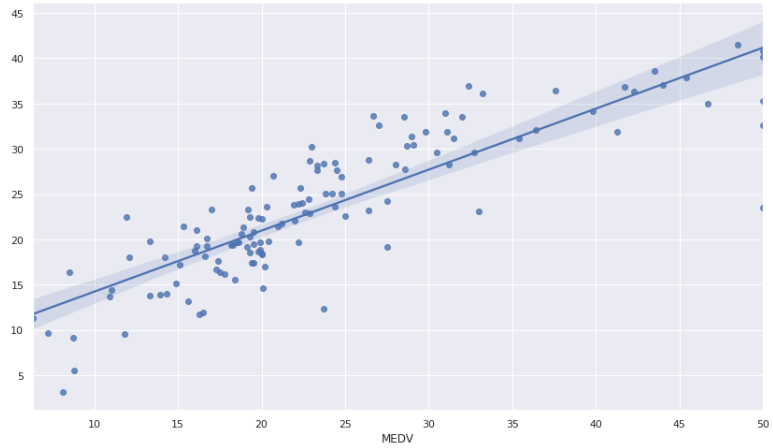

Figure 21: The true values (in points) against the predicted values of our model (linear line).

project is Ridge. Compared to OLS, it gives the similar values for the metrics test MSE and R^2 score but a bigger one for training MSE. Once again, when we combine this with Bootstrap, bias-variance trade-off shows the same behavior like in OLS and also this relation is demonstrated in terms of various values of tuning parameter λ. That is as λ increases, the Ridge regression model becomes simpler, leading to a decrease in variance but an increase in bias. Consider the combination of Ridge and CV, Scikit-learn packages are used to verify the change of MSE with different λ. From this, we can easily choose the value for tuning parameter λ giving the best model for our data. All

the methods are then applied for the Boston Housing dataset. In the end, we find out the model in which polynomial degree is 2 combining with Ridge regression is so far the best. This is proved by a linear fit between the true and predicted values using our model. By using different number of features to train the model, it was observed that the MSE of the prediction increases as you increase the number of features used. The best combination so far is the model that uses at least 5 different features.

References

[1] Stavros P. Adam, Stamatios-Aggelos N. Alexandropoulos, Panos M. Pardalos, and Michael N. Vrahatis. No free lunch theorem: A review. May 2019.

[2] Bradley Boehmke and Brandon Greenwell. *Hands-On Machine Learning with R.* CRC Press, 2019.

[3] Efron Bradley and Robert Tibshirani. *Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy.* Statistical Science, 1986.

[4] M. Hjorth-Jensen. Lecture notes on machine learning, February 2021.

[5] Qshick. Ridge regression for better usage, January 2019.

[6] Qiubing Ren, Mingchao Li, and Shuai Han. Tectonic discrimination of olivine in basalt using data mining techniques based on major elements: a comparative study from multiple perspectives. *Big Earth Data*, 3:7, February 2019.

[7] S. Sharma. Overfitting vs underfitting.

[8] D. Witten, R. Tibshirani, T. Hastie, and G. James. *An Introduction to Statistical Learning: With Applications in R*, volume 1. Springer, June 2013.