PROJECT 1: Linear Regression Analysis and Resampling Methods

Alma Kurmanova
Yash Kumar
UNIVERSITY OF CAEN
Machine Learning Course

February 24, 2021

Abstract

Machine learning is becoming a unique technique in scientific researches, in many sciences, it has replaced the traditional statistical methods. Since machine learning provides simpler and faster techniques to perform the analysis. One of the science used actively machine learning techniques is Physics. For instance, supervised machine learning algorithms to a well-known model in Physics is applied for many pieces of research and the same algorithm was used in the project.

In this work, we focus on the analysis of the Franke Function and a real data "Boston Housing data set" as a regression problem. Ordinary least square and ridge regression are applied to both problems and the predictive model is assessed. In addition, the impact of resampling techniques such as bootstrap and cross-validation on the obtained results are evaluated. Finally, a critical evaluation of all the algorithms used is presented, with a discussion about their respective performance and usability in cases considered.

Contents

1 Introduction

Machine learning (ML) is the study of computer algorithms and statistical models that computer systems used to learn and improve their performance of a task on the basis of their own previous experience. This collection of statistical methods has already proved to be capable of considerably speeding up both fundamental and applied research. For example, it can be used to identify the risk factors for prostate cancer, based on clinical and demographic variables.

The example above describe the supervised learning problem, named due to the presence of both raw input and as well as results. Thus, the training data set is used to train a model whereas the test data set is new data for predicting results or to see the accuracy of the model.

There are two major types of supervised machine learning algorithms:

- *Classification*: The goal is to predict a class label, which is a choice from a predefined list of possibilities. Classification is divided into binary classification, which is the special case of distinguishing between two classes, and multi-class classification, which is classification between more than two classes.

- *Regression*: In this case, the goal is to predict a continuous number or a floating point number in programming terms.

The main aim of the present work is to study regression methods such as Ordinary Least Square fitting (OLS) and Ridge in detail while also using resampling techniques like the bootstrap method and cross validation. The methods will be first applied to the Franke function for a theoretical analysis, followed by a real-world application in the form of the Boston housing data set, as will be demonstrated in the following sections.

The first part of the report is devoted to the theoretical background of OLS and Ridge regression methods, resampling approaches and bias-variance tradeoff. This is followed by the section on the methodology for

The second part of the work examines the application of the above-mentioned approaches and methods to the Franke function and Boston housing data and presents the results and conclusion.

2 Theoretical Background

The methods have three main steps in general. The first is considered to prepare the data set based on the division of the data set into training and test data. The next step is a model development designed by the input and output of the training data, which reflect the features of the system. The last constituent allows estimating how good the model is in reproducing the data, and can be done in multiple ways, but one chiefly deals with minimization of a cost function.

2.1 Data set preparation

In machine learning, a main task is to study and create algorithms based on the learning of data and making predictions or decisions [1], through building a mathematical model from input data. The data used to create a model are commonly from the multiple data sets. In particular, three data sets are commonly used in the different stages of the model development: training, validation and test data sets.

The training data is a data set used in the learning stage and used to fit the parameters of the model. In practice, the training data often is pairs of an input vector (or scalar) and the corresponding output vector (or scalar), where the answer key is commonly denoted as the target (or label). The output resulted from the model is compared with the actual data set. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted.

The validation data set is a sample of data used to provide an unbiased evaluation of a model fit on the training data set while tuning model hyperparameters.Validation can be used for the regulation of early stopping (stopping training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset). This simple procedure is complicated in practice by the fact that the validation data set's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when overfitting has truly begun [2].

The test data set is used to provide an unbiased evaluation of a fitted model on the training data set. The term "validation set" is sometimes used instead of "test set" in some literature (e.g., if the original dataset was partitioned into only two subsets, the test set might be referred to as the validation set)[3].

In the following section a regression, in particular the ordinary least squares and Ridge regression, and resampling approaches as bootstraping and cross-validation will be described.

2.2 Model development

In the project the linear methods for regression are used as the approach to model the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variable). Linear regression models are fitted using the ordinary least squares,

but it can be fitted by Ridge regression to avoid overfitting. More details about these two methods will be given in the following sections.

2.2.1 Ordinary Least Squares

Ordinary least squares, or linear least squares, is a method that estimates the parameters in a regression model by minimising the sum of the squared residuals. This method draws a line through the data points that minimises the sum of the squared differences between the observed values and the corresponding fitted values.

Linear Regression is a statistical model, widely used in ML regression tasks, based on the idea that the relationship between two variables can be explained by the following formula:

$$y = \beta_0 + \beta_i x_i + \epsilon_i \tag{1}$$

Where ϵ_i is the error term, and β are unknown parameters or coefficients, and the variables X_i.

The idea of Simple Linear Regression is finding those parameters β for which the error term is minimised. To be more precise, the model will minimise the squared errors.

$$RSS(\beta) = \sum_{i=1}^{n}(y_i - f(x_i))^2 = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \tag{2}$$

Once obtained the value of βwhich minimise the squared errors, the model's equation will have the following form:

$$\hat{y} = \hat{\beta}x \tag{3}$$

in order to obtain the infamous OLS parameter estimates

$$\beta = X' \tag{4}$$

$$\beta^{OLS} = (X'X)^{-1}(X'Y). \tag{5}$$

However, simple linear regression can lead to over-fitting. Thus, there are some of the simple techniques to reduce model complexity while preventing over-fitting, such as Ridge and LASSO regression. Ridge regression will be described in detail in the following section.

2.2.2 Ridge Regression

Ridge regression is a model tuning method that is used to analyse any data that suffers from multicollinearity. This method performs least-squares regularization. When the issue of multicollinearity occurs, least-squares are unbiased, and variances are large, this results in predicted values to be far away from the actual values.The ridge coefficients minimize a penalized residual sum of squares,

$$\hat{\beta}_{ridge} = argmin \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{6}$$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage. The coefficients are shrunk toward zero (and each other). The idea of penalizing by the sum-of-squares of the parameters is also used in neural networks, where it is known as weight decay. Writing the criterion in matrix form,

$$RSS(\lambda) = (y - \mathbf{X}\beta)^T (y - \mathbf{X}\beta) + \lambda \beta^T \beta \tag{7}$$

the ridge regression solutions are easily seen to be

$$\hat{\beta}_{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^T y \tag{8}$$

where $\mathbf{I}$ is the pxp identity matrix. The solution adds a positive constant to the diagonal of $\mathbf{X}^T\mathbf{X}$ before inversion. This makes the problem nonsingular, even if $\mathbf{X}^T\mathbf{X}$ is not of full rank, and was the main motivation for ridge regression.

2.3 Model assessment and selection

2.3.1 Resampling approaches

Resampling technique is used for the following purposes:

- estimating the precision of sample statistics (medians, variances, percentiles) by using subsets of available data (jackknifing) or drawing randomly with replacement from a set of data points (bootstrapping);
- exchanging labels on data points when performing significance tests (permutation tests, also called exact tests, randomization tests, or re-randomization tests)
- validating models by using random subsets (bootstrapping, cross validation).

Two of the most commonly used resampling methods are presented: cross-validation and the bootstrap.

Bootstrap

Bootstrapping is a statistical method to estimate the distribution of an estimator by sampling with replacement from the original sample, often to derive robust estimates of standard errors and confidence intervals of a parameter like a mean, median, proportion, odds ratio, correlation coefficient or regression coefficient.

Cross Validation

Statistical method for validation of a predictive model is cross-validation model. The model considers to held out one subset of the data for using as validation set, remaining part of the training data is used to predict and fit the model for the validation set. Averaging the quality of the predictions across the validation sets yields an overall measure of prediction accuracy.

2.3.2 Bias–variance tradeoff

In statistics, the bias and the variance are two critical characteristics of estimators. The bias is the difference between the true population parameter and the expected estimator and it measures the accuracy of the estimates. The variance measure the spread of these estimates.

In statistics and machine learning, the bias–variance tradeoff is the property of a set of predictive models whereby models with a lower bias in parameter estimation have a higher variance of the parameter estimates across samples, and vice versa. The bias–variance dilemma or bias–variance problem is the conflict in trying to simultaneously minimise these two sources of error that prevent supervised learning algorithms from generalising beyond their training set. The bias-variance tradeoff is a central problem in supervised learning. Ideally, one wants to choose a model that both accurately captures the regularities in its training data, but also generalises well to unseen data. Unfortunately, it is typically impossible to do both simultaneously. High-variance learning methods may be able to represent their training set well but are at risk of overfitting to noisy or unrepresentative training data. In contrast, algorithms with high bias typically produce simpler models that don't tend to overfit but may underfit their training data, failing to capture important regularities.

Models with high variance are usually more complex (e.g. higher-order regression polynomials), enabling them to represent the training set more accurately. In the process, however, they may also represent a large noise component in the training set, making their predictions less accurate – despite their added complexity. In contrast, models with higher bias tend to be relatively simple (low-order or even linear regression polynomials) but may produce lower variance predictions when applied beyond the training set [4].

3 Methodology

The main aim of this project is to study regression problems with Franke's function and real data.

For the first part of the exercise, the data for Franke's function is used to study fitting polynomials to a specific two-dimensional function called Franke's function. This is a function that has been widely used when testing various interpolation and fitting algorithms. First, the data set x and y in $[0, 1]$ with 0.05 step and required arrays have been produced and split using *Scikit Learn library*. At the next step, OLS regression has been used for modeling it and calculate R^2, MSE, variance, and bias. For comparison, the data have been split and scaled with *Standard scaller* and *Minmax scaller* from Scikit learn library. R^2, MSE, bias, and variance have been calculated with obtained scaled data and compared with previous results for unscaled data.

The Franke function, which is a weighted sum of four exponentials reads as follows:

$$f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) + \\ + \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right).$$

The second part of the project considers using the resampling technique as bootstrapping. As the function, Franke's function has been used and a linear regression has been done with ordinary least squares. After splitting data into training and testing one's regression with bootstrap has been performed in order to perform a proper assessment of a model. Furthermore, after the above-mentioned processes, the Bias-Variance trade-off has been studied in detail.

Thereafter the next part of the exercises involves the resampling technique as cross-validation. To assess how the scaling before the whole procedure, or during the procedure in/between each splitting can be affected on the result, it was checked to exclude data leakage. The following step considers the implementation of the k-fold cross-validation algorithm and evaluation again the MSE function resulting from the test folds and compared with the results from bootstrap.

For the next data set Ridge regression has been used from Scikit-Learn functionality and both resampling techniques as bootstrap and cross-validation have been a part of the analysis for different values of λ. The obtained result have been compared with results from the first two parts of the exercise to study the dependence on λ. Additionally, the bias-variance trade-off as a function of various values of the parameter λ has been studied to assess a model.

For the final part of the project a real data "Boston Housing data set", which contains 13 features, and 1 target named 'MEDV'. The objective of the exercises is to perform the polynomial regression using both OLS and Ridge Regression in order to find the model that best predicts the 'MEDV' target value. To assess the quality of a model MSE and the R2 score have been used.

In order to choose high correlated features with the target the pair plot from *Seaborn library* has been plotted and from the [-1,1] features with [-0.45,0.45] correlation with the target have

been excluded. The remaining features have been checked for the high correlation with each other. The last steps have been done manually using the obtained correlation matrix. Splitting the data, design matrix and OLS and Ridge regression have been applied into real data including chosen features and the target data. The final task of the exercise is to find the best model, which is achieved by varying λ parameters and complexity.

3.1 Python code

All the code in which this report is based has been written in Python 3.0. The .py files can be found in the folder called 'CODE', available in https://github.com/AlmaNucPhysics/Project-1:

- Project1.1: Ordinary Least Square (OLS) on the Franke function;
- Project1.2: Bias-variance trade-off and resamplng techniques;
- Project1.3: Cross-validation as resampling techniques, adding more complexity;
- Project1.4: Ridge Regression on the Franke function with resampling;
- Project1.5: Introducing Real Data.

Further description is included in the comments along the code. A RESULTS folder with the main results is contained in the same repository.

4 Results: Result and Analysis

In this section the results of the code created for this project are presented and explained in detail.

4.1 Part a: Ordinary Least Square on the Franke function

The data set for a a function FrankeFunction(x, y) with $x, y \in [0, 1]$ has been generated and the noise with the normal distribution $N(0, 1)$ has been added to the function $f(x, y)$. The design matrix has been written to perform the ordinary least regression analysis with the polynomial order up to five. The Franke's function has been plotted with using of MatPlotlib library and presented (see 1).

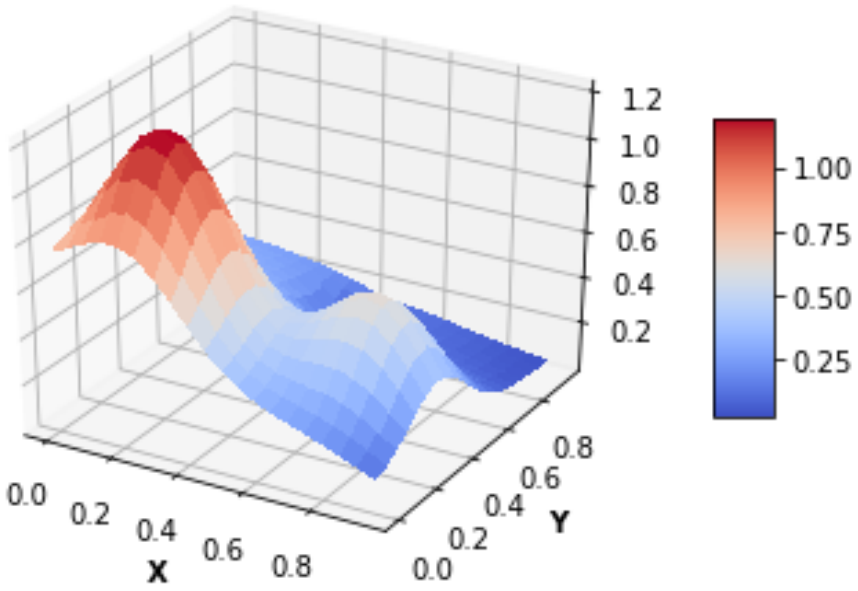


Figure 1: Franke's Function

The data for training and test have been splitted by the Scikit Learn library and the scaling techniques like Standard and MinMax approaches have been applied to obtained data before the regression.

The evaluation of the Mean Squared error (MSE) has been calculated with the following formula:

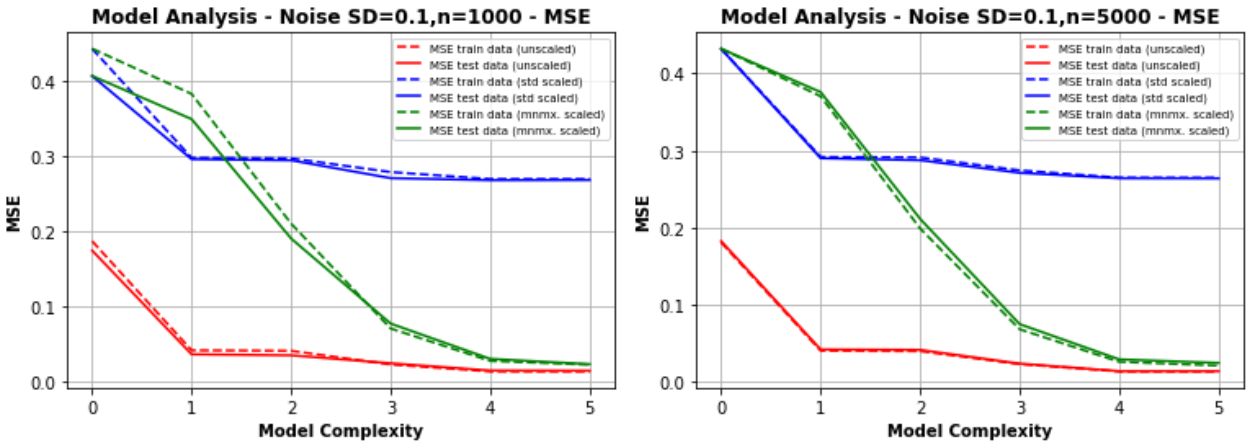
$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=0}^{n-1} (y_i - \tilde{y}_i)^2,$$

and the R^2 score function. If $\tilde{\boldsymbol{y}}_i$ is the predicted value of the $i - th$ sample and y_i is the corresponding true value, then the score R^2 is defined as

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

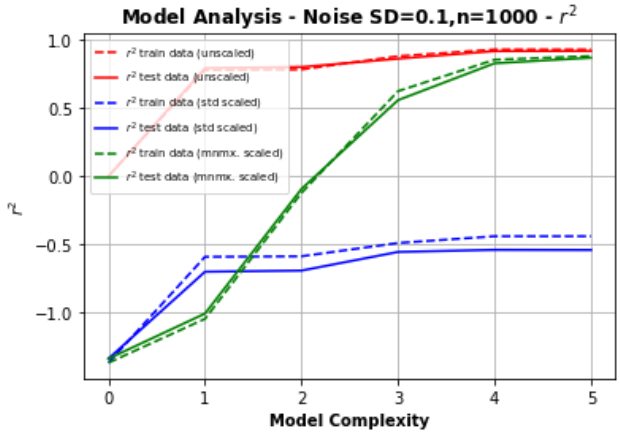
where we have defined the mean value of $\boldsymbol{y}$ as

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

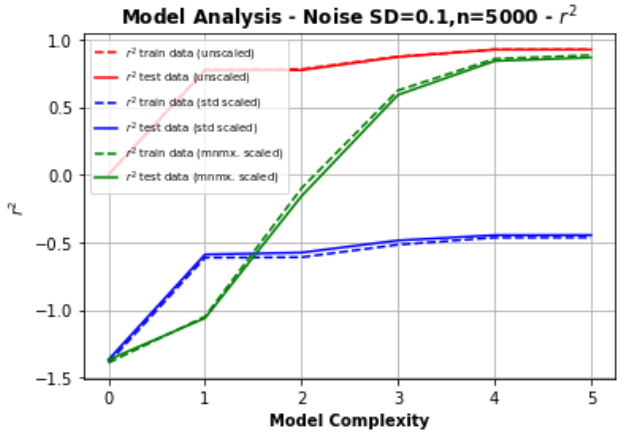


(a) MSE, noise=0.1, n=1000

(b) MSE, noise=0.1, n=5000



(c) R^2, noise=0.1, n=1000



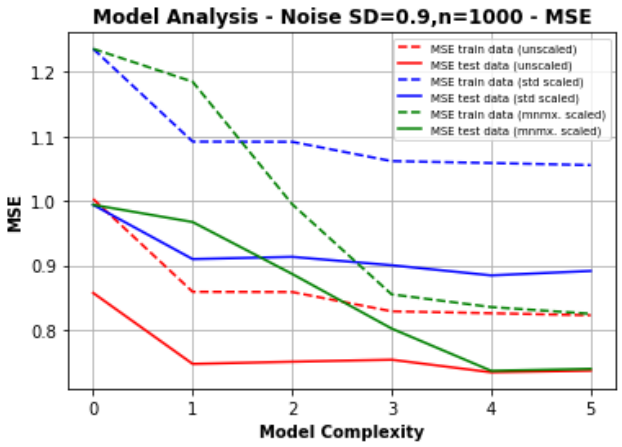
(d) R^2, noise=0.1, n=5000

Figure 2: Variation of model performance with number of data points

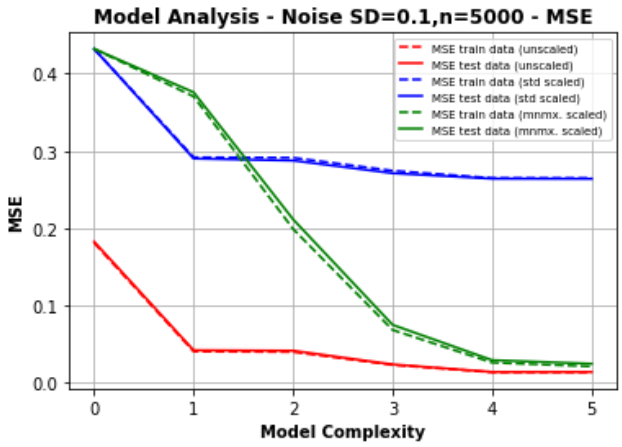
In order to compare the influence of the data points number on the MSE results, two graphs are presented (see figure 2). Noise added to the function has the standard deviation of 0.1 for both data sets. Increasing the numbers of data points provides a better model for the prediction of test data, hence the difference between the test and training data is no significant. However, scaled and unscaled data have significant divergences. MSE results of unscaled data present a better outcome compared with the scaled data since MSE should be as small as possible.

To identify the effect of the standard deviation of the noise 0.9 and quantity of the data points on MSE calculation standard deviation has been altered for data with 1000 and 5000 points (see figure 3). The standard deviation of 0.9 make MSE for the training and test data diverge significantly for 1000 point than for 5000 points. Since it provides more data to train

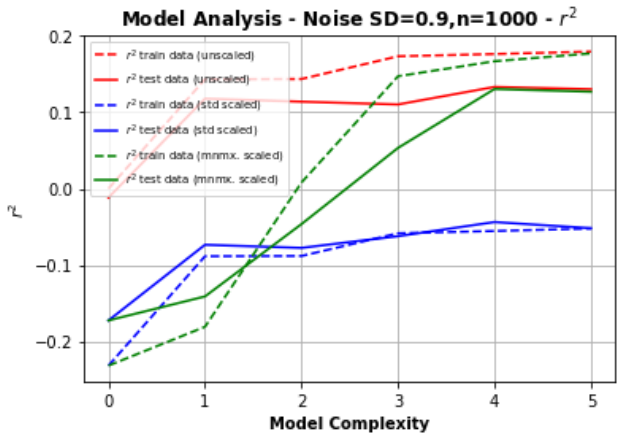
and predict it. Although standard deviation has been changed, unscaled MSE is still better compared to scaled data.



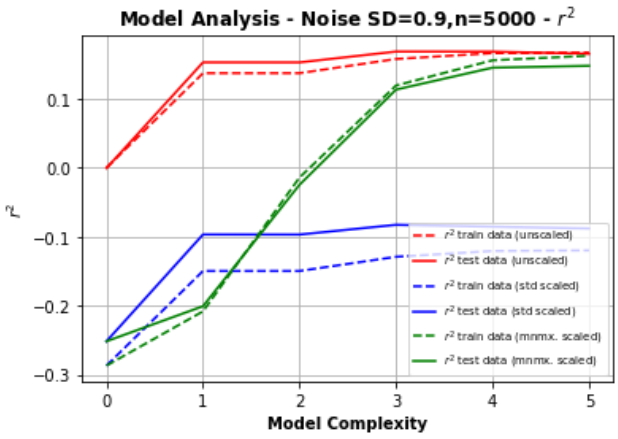
(a) MSE, noise=0.9, n=1000



(b) MSE, noise=0.9, n=5000



(c) R^2, noise=0.9, n=1000



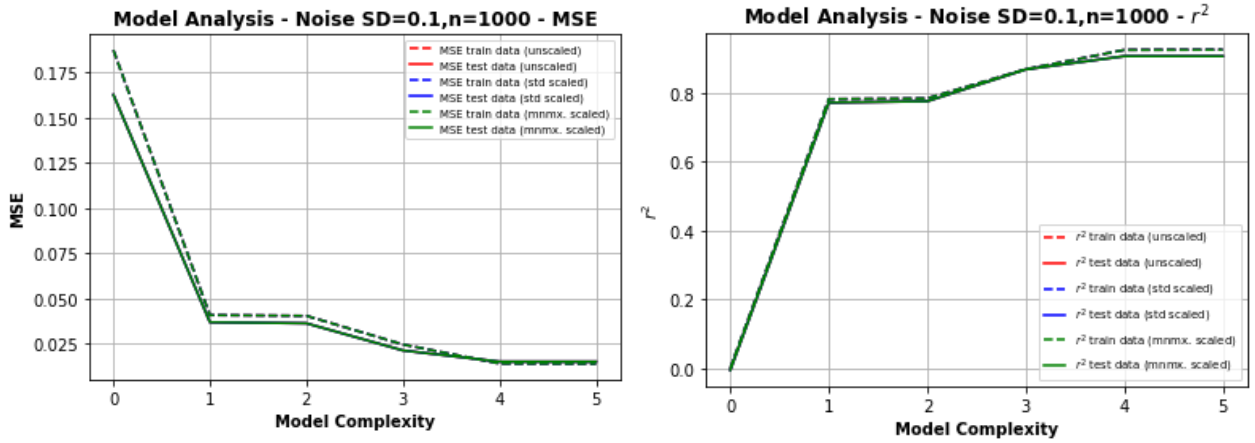
(d) R^2, noise=0.9, n=5000

Figure 3: Variation of model performance with number of data points

For standard and MinMax scaled, and unscaled data R^2 have been calculated with 0.1 standard deviation for data point 1000 and 5000 (see figure 2).Unscaled data are shown better results. However, MinMax scaled data are achieved at the same level as unscaled data for model complexity= 4. The graph R^2 for unscaled data vs. model complexity presents the sufficiency of the model in comparison with the scaled data vs.model complexity, since R^2 should be approximately 1. R^2 calculation for 1000 and 5000 points the difference between training and test data have not been observed. Nevertheless R^2 with a standard deviation of 0.9 a number of data points reflects the remarkable difference.

Self-written algorithms based on pseudoinverse of design matrix showed objectively worse performance in case of scaled data (see figure 2 and 3), but when the fitting was performed using the regression functionality of scikit-learn, both scaled and unscaled data showed same behaviour of R^2 and MSE with respect to model complexity (figure 4). This could only be

obtained by setting the *Fit Intercept* parameter to *True*, the presence of a intercept column in the design matrix notwithstanding. Since the self-written algorithm did not contain any analogue of the *Fit intercept* function, it always resulted in worse performance on part of scaled data, which could also be seen by setting *Fit intercept* to *False*.



(a) MSE, noise=0.1, n=1000

(b) R^2, noise=0.1, n=5000

Figure 4: MSE and R^2 with noise=0.1 and n=1000 and 5000 when regression is performed with Scikit-Learn

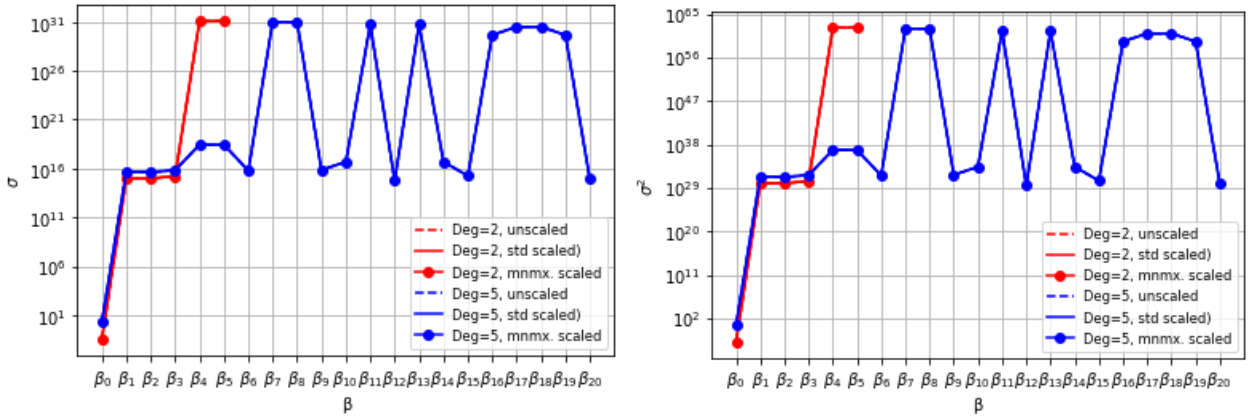
The variance of the model parameters β are simply the diagonal elements of the matrix $(X^TX)^{-1}$. This expression is, however, valid only if the square matrix X^TX is not singular, which was not the case here. With singular matrices, one needs to proceed with a full SVD procedure, as detailed in [5]. A code based on the same algorithm was written for confidence intervals of the β parameters and variance of the model parameters for Franke's function, which provide bad representation (see figure 5). However, the same code was checked with another function and a satisfactory result had been obtained.

4.2 Part b: Bias-variance trade-off and resampling techniques

The part of the project focuses on implementing the resampling technique as bootstrap using the Franke's function. In order to create a model OLS is used and including the resampling technique to perform the bias-variance trade-off.

To check the MSEs of training and test data the graph MSE vs model complexity has been presented (see figure 6). High bias and low variance part graph shows and it is possible to observe the divergence between test and training data for higher model complexity, which will reach at the part with low bias and high variance between training and test data. Since aim of the part is to calculate bias, error, and variance for Franke's function with polynomial order up to five. The graph between MSE and model complexity has been plotted with polynomial order up to five.

With this result we move on to the bias-variance trade-off analysis.



(a) Confidence intervals of parameters β

(b) Variance of the model parameters

Figure 5: Confidence intervals and variance of the model parameters

Consider a dataset $\mathcal{L}$ consisting of the data $\mathbf{X}_{\mathcal{L}} = \{(y_j, \boldsymbol{x}_j), j = 0 \dots n-1\}$.
Let us assume that the true data is generated from a noisy model

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

Here ϵ is normally distributed with mean zero and standard deviation σ^2.

In our derivation of the ordinary least squares method we defined then an approximation to the function f in terms of the parameters $\boldsymbol{\beta}$ and the design matrix $\boldsymbol{X}$ which embody our model, that is $\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\beta}$.

The parameters $\boldsymbol{\beta}$ are in turn found by optimizing the means squared error via the so-called cost function

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

Here the expected value $\mathbb{E}$ is the sample value. Let's investigate the expectation inside the sum.

$$\begin{aligned}\mathbb{E}[(y_i - f_i + f_i - \tilde{y})^2] = \mathbb{E}[(y_i - f_i)^2] + \mathbb{E}[(f_i - \tilde{y})^2] + 2\mathbb{E}[(f_i - \tilde{y})(y_i - f_i)] = \mathbb{E}[\epsilon^2] + \\ \mathbb{E}[(f_i - \tilde{y})^2] + 2(\mathbb{E}[f_i y_i] - \mathbb{E}[f_i^2] - \mathbb{E}[\tilde{y} y_i]) + \mathbb{E}[\tilde{y} f_i])\end{aligned} \tag{9}$$

Note:

- $\mathbb{E}[f_i y_i] = f_i^2$ since f is deterministic and $\mathbb{E}[y_i] = f_i$
- $\mathbb{E}[f_i^2] = {f_i}^2$ since f is deterministic
- $\mathbb{E}[\tilde{y} f_i] = \mathbb{E}[\tilde{y}(f_i \epsilon)] = \mathbb{E}[\tilde{y} f_i + \tilde{y}\epsilon] = \mathbb{E}[\tilde{y} f_i] + 0$ (the last term is zero because the noise in the infinite test set over which we take the expectation is probabilistically independent of the NN prediction). Thus the last term in the expectation above cancels to zero.

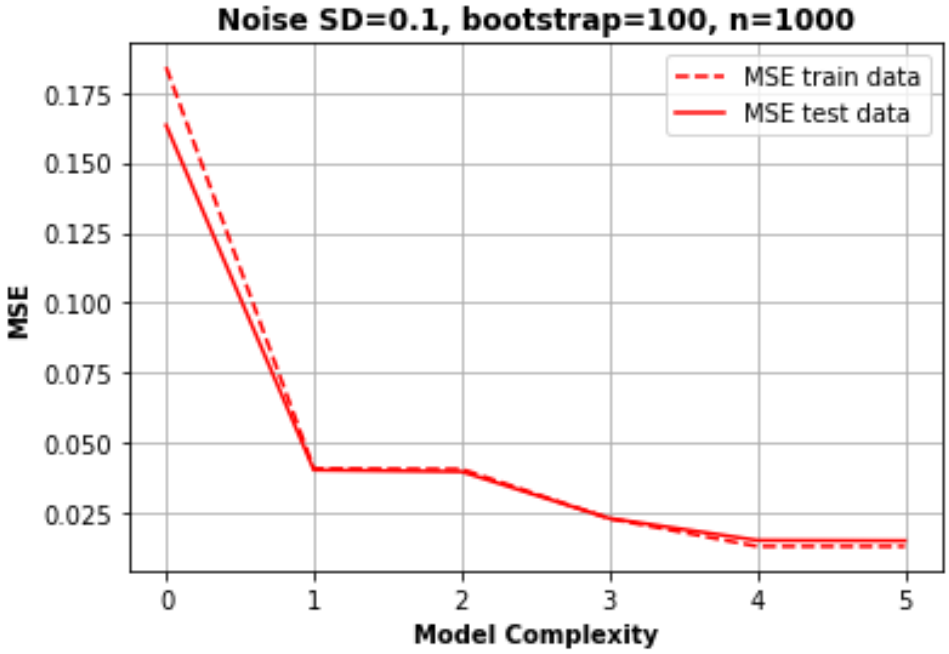


Figure 6: MSE of training and test data, bootstrap=100, n=1000

- $\mathbb{E}[(y_i - \tilde{y})^2] = \mathbb{E}[\epsilon^2] + \mathbb{E}[(f_i - \tilde{y})^2]$

Thus the MSE can be decomposed in expectation into the variance of the noise and the MSE between the true function and the predicted values. This term can be further composed with the same augmentation trick as above.

$$\begin{aligned}\mathbb{E}[(f_i - \tilde{y})^2] = \mathbb{E}[(f_i - \mathbb{E} + \mathbb{E} - \tilde{y})^2] = \mathbb{E}[(f_i - \mathbb{E}[\tilde{y}])^2] + \mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})^2] + \mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})^2] + \\ 2\mathbb{E}[(\mathbb{E}[\tilde{y}] - \tilde{y})(f_i - \mathbb{E}[\tilde{y}])] = bias^2 + Var[\tilde{y}] + 2(\mathbb{E}[f_i\mathbb{E}[\tilde{y}] - \mathbb{E}[\mathbb{E}[\tilde{y}]^2] + \mathbb{E}[\tilde{y}\mathbb{E}[\tilde{y}]]])\end{aligned} \tag{10}$$

Note:

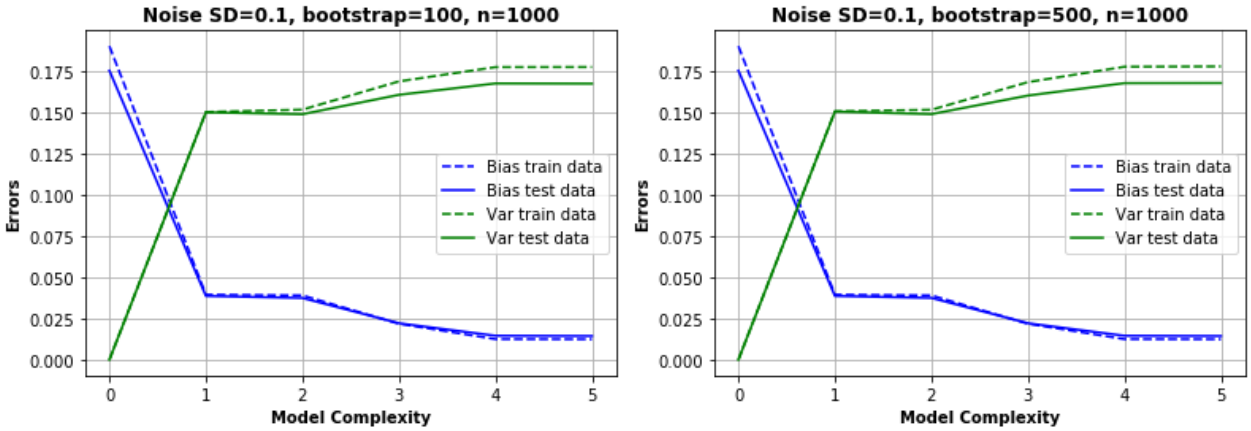
- $\mathbb{E}[f_i\mathbb{E}[\tilde{y}]] = f_i\mathbb{E}[\tilde{y}]$ since f is deterministic and $\mathbb{E}[\mathbb{E}[z]] = z$
- $\mathbb{E}[\mathbb{E}[\tilde{y}]^2] = \mathbb{E}[\tilde{y}]^2$ since $\mathbb{E}[\mathbb{E}[z]] = z$
- $\mathbb{E}[\tilde{y}f_i] = f_i\mathbb{E}[\tilde{y}]$
- $\mathbb{E}[\tilde{y}\mathbb{E}[\tilde{y}]] = \mathbb{E}[\tilde{y}]^2$
- Thus the last term in the expectation above cancels to zero

$$\mathbb{E}[(f_i - \tilde{y})^2] = bias^2 + Var(\tilde{y})$$

Thus the decomposition of the MSE in expectation becomes:

$$\mathbb{E}[(y_i - \tilde{y})^2] = Var(noise) + bias^2 + Var(y_i)$$

A bias-variance analysis of the Franke's function has been performed by studying the MSE value as function of the complexity of your model. To identify the influence of the number of bootstrap on the error of training and test data, 100 and 500 bootstrap have been applied into data (see figure 7). The bootstrap method involves resampling a dataset with replacement. Actually, the effect of the bootstrap method in the bias-variance trade-off is almost negligible for the linear regression models applied in this study.



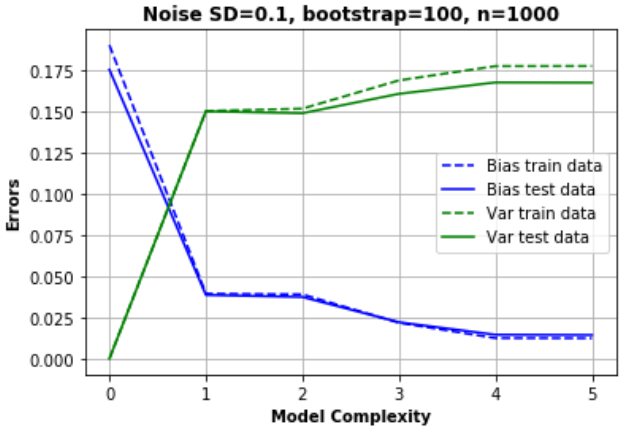
(a) SD=0.1,bootstrap=100,$n_{data} = 1000$ (b) SD=0.1,bootstrap=500,$n_{data} = 1000$

Figure 7: Bias and variance trade-off analysis with SD=0.1,bootstrap=100 and 500,$n_{data} = 1000$

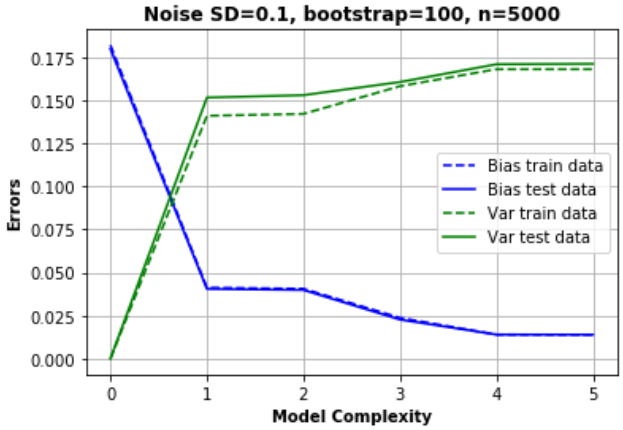
From the figure 7 it is clear that the increasing the bootstrap number have no impact on the results significantly.

To evaluate the dependence the bias and variance trade-off on data points, 1000 and 5000 points have been used to plot the error vs model complexity (see figure 8).As mentioned before one of the objectives is the evaluation of the effect of data points number on the results of bias, variance trade-off analysis. For the comparison, the number for data points has been chosen as 1000 and 5000. It is expected to get a better predictive model with a higher number of data points, which is observed in the graph. The bias for the training and test data for $n = 1000$ at lower model complexity has been observed, but for $n = 5000$ this discrepancy has need disappeared. Concerning the variance it shows for $n = 1000$ low value up to 2 polynomial order, then the variance is increasing between the training and test data. In case $n = 5000$, increasing of the variance has been seen only between $1 - 3$ polynomial order (in case,up to five polynomial order), the rest part of the graph presents approximately the same difference between the training and test data. Hence, the impact of the number of the data point is required to study in detail and for higher polynomial order.

The next step is to check noise parameter effects on bias and variance trade-off analysis and it is presented in figure 9. It can be seen higher noise SD give higher error, because it is calculated with bias and variance. Whereas the figure 8a presents lower error.

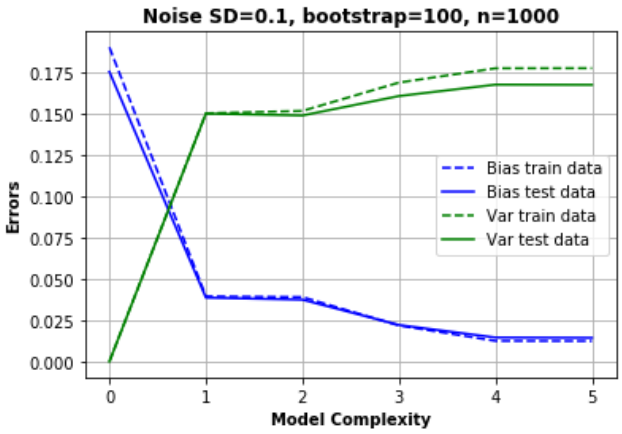


(a) SD=0.1,bootstrap=100,n_{data} = 1000

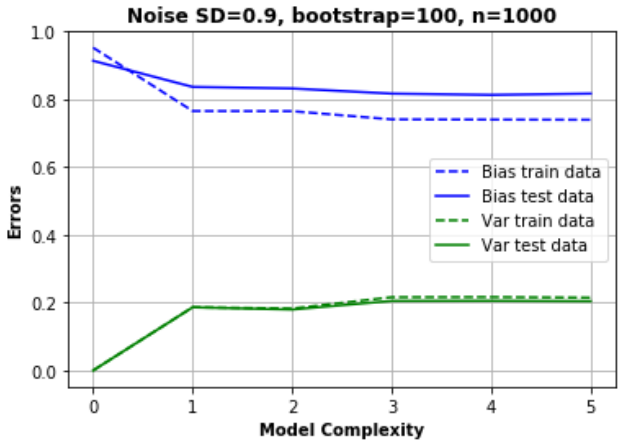


(b) SD=0.1,bootstrap=100,n_{data} = 5000

Figure 8: Bias and variance trade-off analysis with SD=0.1,bootstrap=100,n_{data} = 1000 and 5000



(a) SD=0.1,bootstrap=100,n_{data} = 1000



(b) SD=0.9,bootstrap=100,n_{data} = 1000

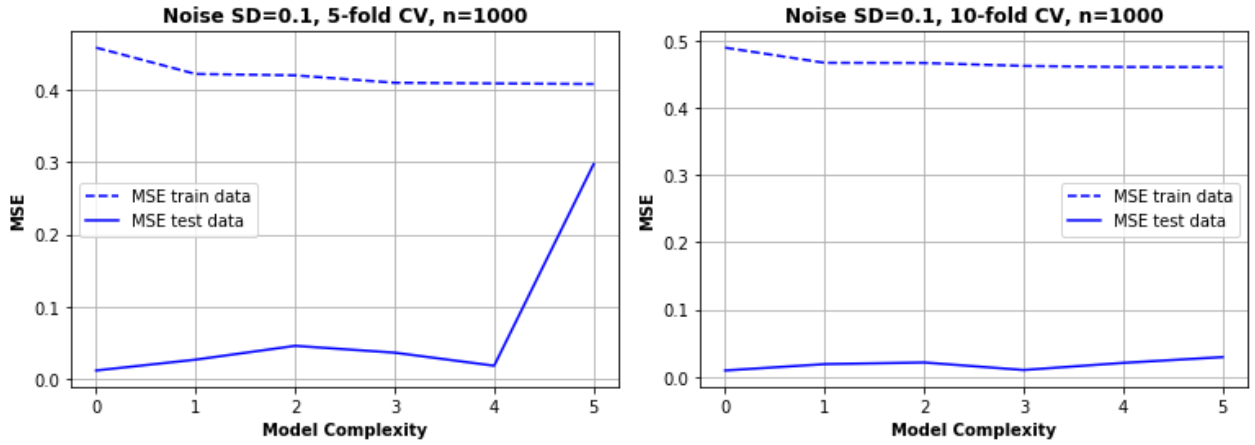
Figure 9: Bias and variance trade-off analysis with SD=0.1 and 0.9,bootstrap=100,n_{data} = 1000

4.3 Part c: Cross-validation as resampling techniques, adding more complexity

In this part of the project, the main objective is to use the resampling technique like cross-validation applied to the Franke function.

The data has been split using of Scikit learn library and scaled with the standard scaler from the same library. In order to evaluate the influence of implementation order between splitting, scaling and cross-validation technique it has been checked and did not reveal any large discrepancies. It is important since the assessment of the effect of order helps to eliminate leakage problem, which can lead to inflated model performance in the lab, and poor performance when deployed with real-world data.

In K Fold cross-validation, the data is divided into k subsets, in this case, k is $5-10$. Now the holdout method is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get the total effectiveness of our model. As can be seen, every data point gets to be in a validation set exactly once and gets to be in a training set k-1 times. This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in the validation set. Interchanging the training and test sets also adds to the effectiveness of this method.



(a) SD=0.1,CV=5,$n_{data} = 1000$ (b) SD=0.9,CV=10,$n_{data} = 1000$

Figure 10: MSE with $5-$ and $10-fold$ CV approach,SD=0.1,bootstrap=100,$n_{data} = 1000$

The cross-validation approach with 5-fold and 10-fold has been implemented to the data and it has revealed that the increasing k-fold value make the graph smoother, which can be seen in figure 10. The variance between training and test data for CV 10-fold is lower compared with CV 5-fold MSEs for the same data. Since CV is used to obtain lower MSE, hence the $k = 10$ provides better result.

Another objective of the exercise is to compare the MSE from CV and bootstrapping codes, which is presented in figure 11. From the graph, the bootstrap approach provides low variance between training and test data, but MSE for the testing data with the CV technique presents higher variance. Since, MSE of test data for CV provides lower value, hence CV performs better output compared with bootstrap.

4.4 Part d: Ridge Regression on the Franke function with resampling

Ridge regression method has been used to predict a model using Scikit learn library.

In order to carry on the previous analysis Ridge regression has been employed instead of OLS and like as in the part b and the part c, the Ridge regression has been combined with

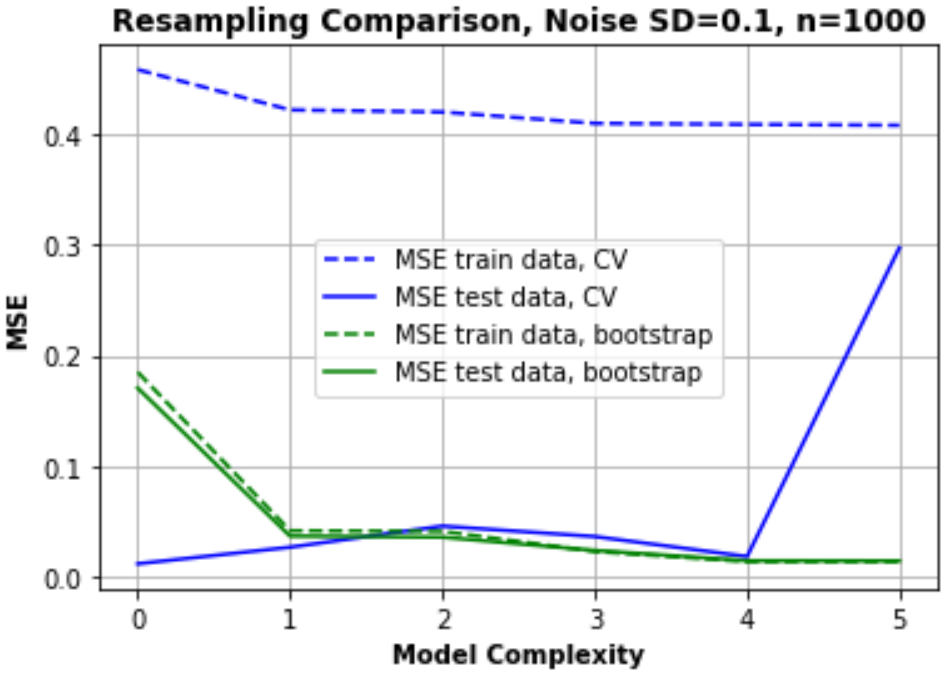


Figure 11: Resampling comparison between CV and Bootstrap approaches

bootstrap and cross-validation resampling techniques for different values of λ to compare and analysis obtained results for each of them.

Bootstrap with $n = 500$ and $10 - fold$ cross-validation analysis have been performed for different values of λ 10^{-2} and 10, which presents in figure 12. It can be seen that MSEs for the training and test data using CV approach show higher value compared with the data using bootstrapping technique. The last data provides a lower value enough, but it is not the only feature to choose the best predictive model. For this reason, the R^2 values for CV and bootstrap have been estimated with λ with 5 and 10. In the figure 13 it can be observed clearly that R^2 values for bootstrap provide better performance compared with CV values. Since the R^2 value should tend to 1, which are got only for values using the bootstrap approach. The best R^2 are in the range of polynomial order $[3, 5]$ and the polynomial order of 4 has been chosen for the following analysis of bias-variance trade-off as a function of various values of the parameter λ. It has been plotted in figure 14. From the figure, λ of 10^2 is considered to be the best parameter for the predictive model.

4.5 Part e: Introducing Real Data

In the part of the project a real data of Boston Housing data set has been used to perform polynomial regression, using both OLS and Ridge regression to find the best model for data prediction, in Boston housing case, the target data is MEDV. MSE and R^2 have been calculated to assess the quality of a model.

The target data depends on 13 features, which causes a problem to create a design matrix.

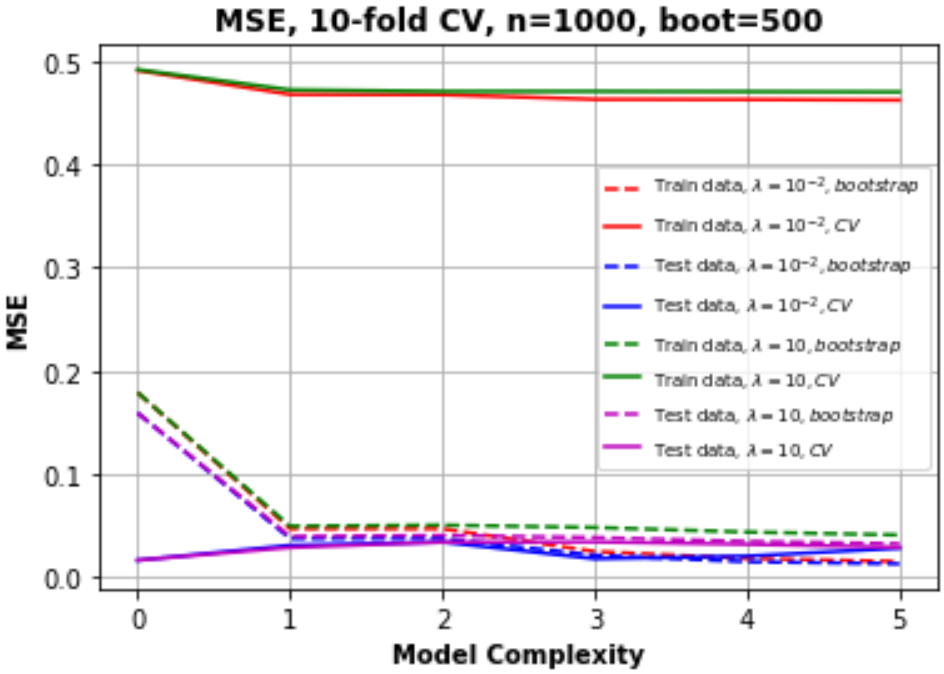


Figure 12: MSE vs Model complexity for CV and bootstrap

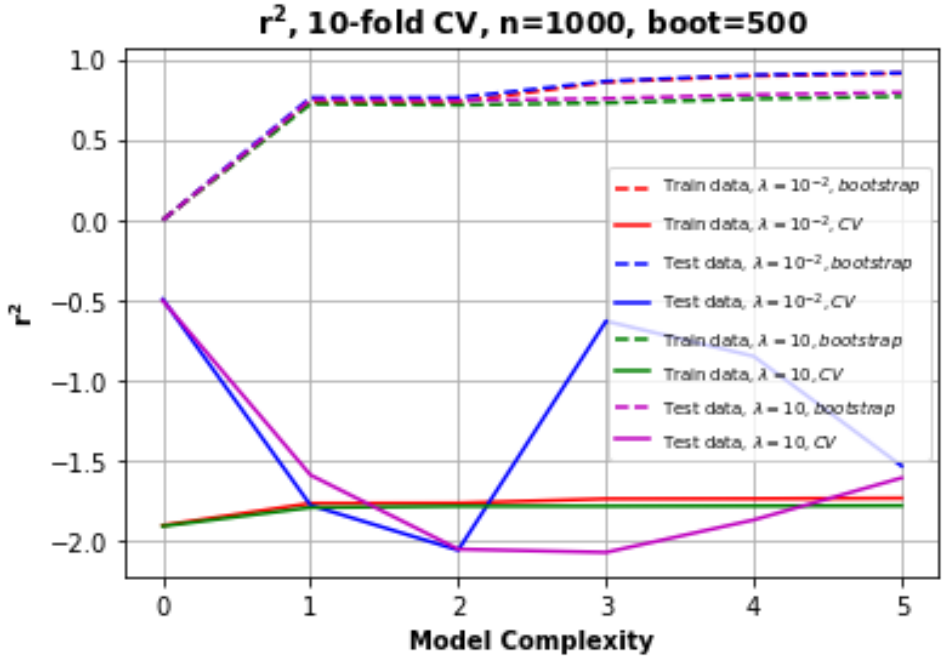


Figure 13: R^2 vs Model complexity for CV and bootstrap

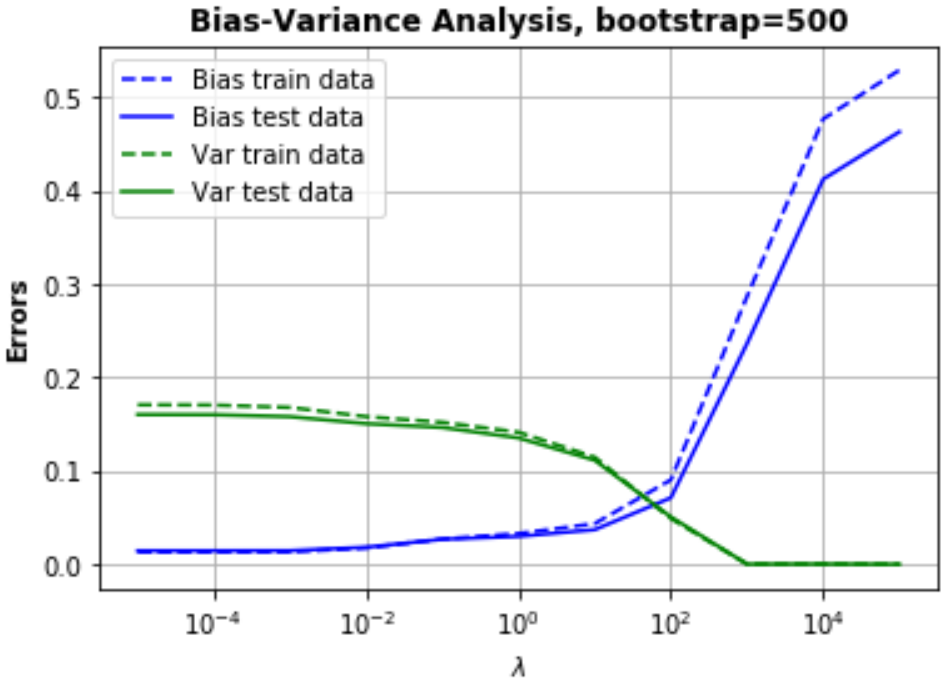


Figure 14: Bias-Variance Analysis

For a reason, the Scikit-Learn's functionality has been used to create a design matrix. Before the design matrix creation, the features number has been reduced using the corrected matrix to choose the most correlation between features and the target. Moreover, it is needed to check for the correlation between the features and exclude one of the features, which is correlated with another one to avoid multicollinearity.

In the figure 15 the correlation between 13 features and the target data are presented. In order to reduce some features from the list, the correlation range $[-0.5, 0.5]$ has been chosen. After the checking of the list, there are 5 features: INDUS, RM, TAX, PTRATIO,and LSTAT. However, some of them are well-correlated with each other, which requires to choose one from two features. From figure 16, LSTAT feature was initially chosen as a good feature based on its strong correlation with MEDV, but LSTAT was also correlated with RM and INDUS. Thus LSTAT was excluded. Similarly, based on its correlation with INDUS and PTRATIO, the feature TAX was excluded from the design matrix as well. Ultimately, we decided to construct a model with 3 features - RM, INDUS, and PTRATIO.

The best model was selected using a sequential check wherein different aspects were optimised one by one. As discussed, the first thing to do was evaluate the number of features to be used, using the parameter correlation heat maps. The obvious next step was determining the complexity of the model which in this case meant determining the maximum degree of the polynomial to be employed. The performance was to be judged through the usual MSE and r^2 values, for regressions made using both OLS and Ridge. This allowed us to simultaneously study the variation of model performance with the regularisation parameter λ and obtain

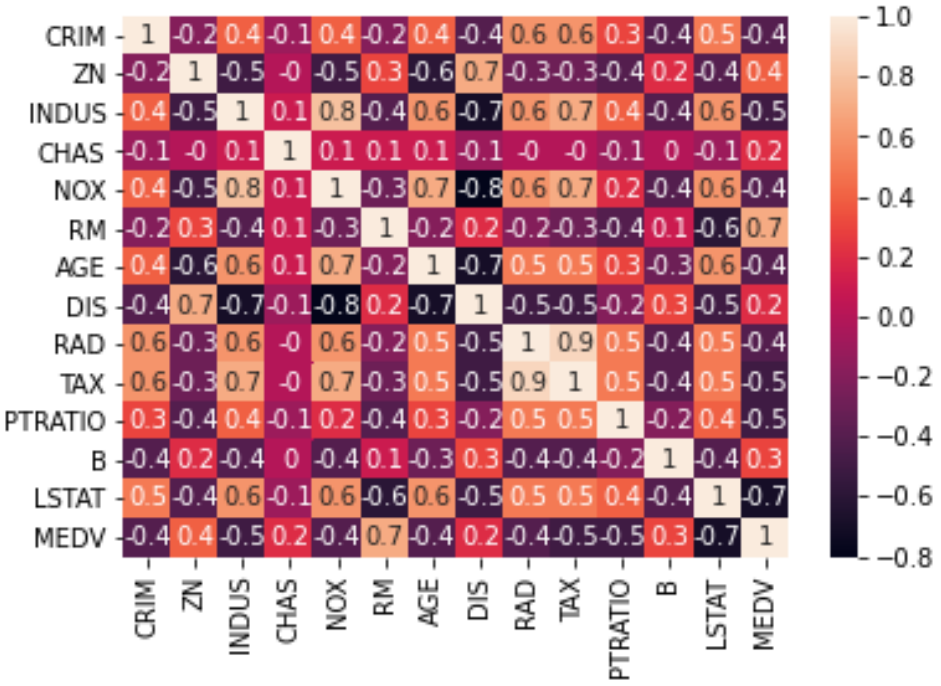


Figure 15: Correlation matrix

insights into the nature of the design matrix itself.

From the figure 17 it can be seen that the performance is becoming worse with increasing of the order of polynomial and the order of 7 can be safely excluded. Amongst orders 3 and 4, the former is a better choice, because it predicts same performance by both Ridge and OLS at low value of λ as expected, unlike in order 4, where the results are different. This tells us that a more complex design matrix has linearly dependent columns since Scikit-Learn forgoes the usual $(X^T X)$ method in favour of *pseudoinverse* in case of OLS, while Ridge proceeds as always with regularisation. Hence, model of order 3 with 3 features was chosen as an optimal model and OLS was selected as a regression.

Finally, cross-validation was applied to chosen model to assess the effect of resampling, but as can be seen from figure 18 the r^2 value is negative, as was also observed with the Franke function.Clearly, cross-validation is not the best resampling approach for small data and perhaps should be replaced with bootstrap.

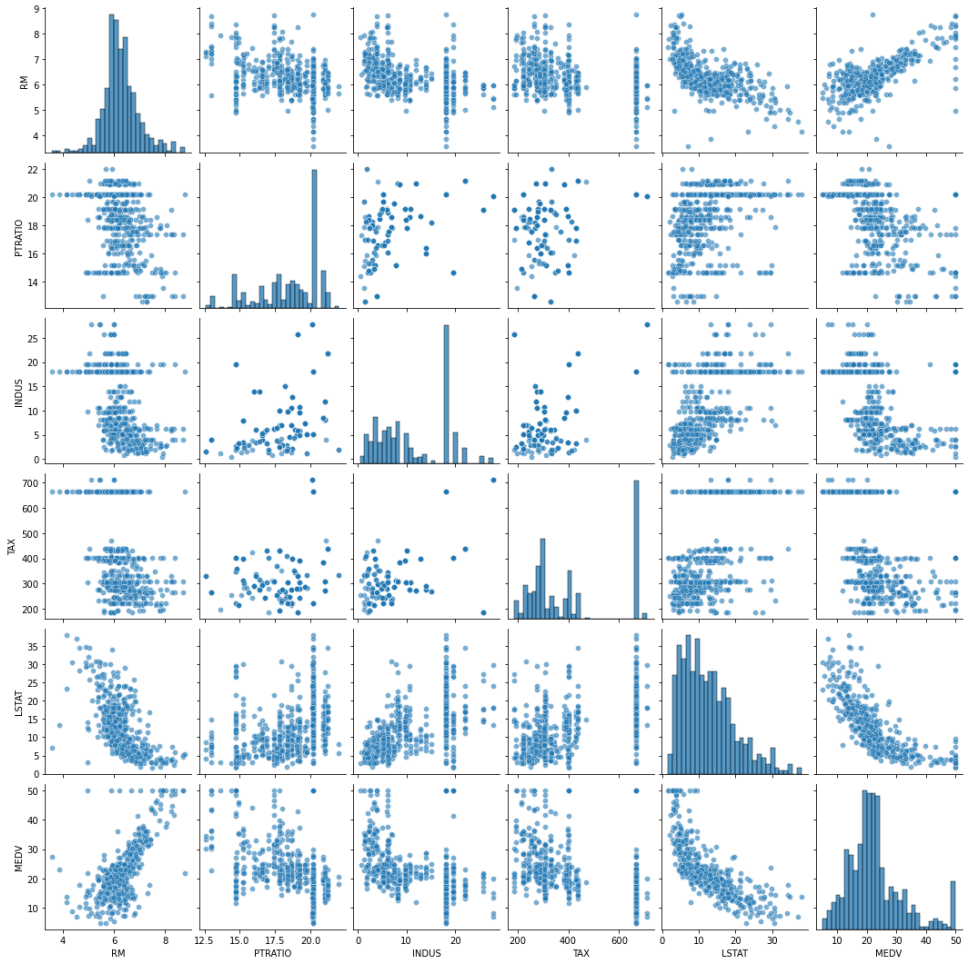
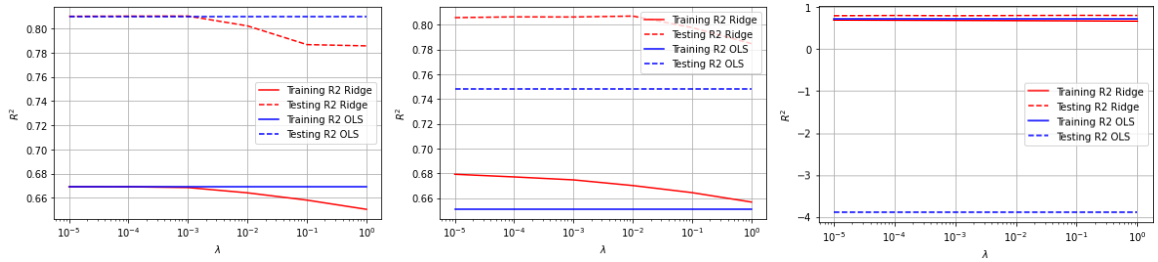
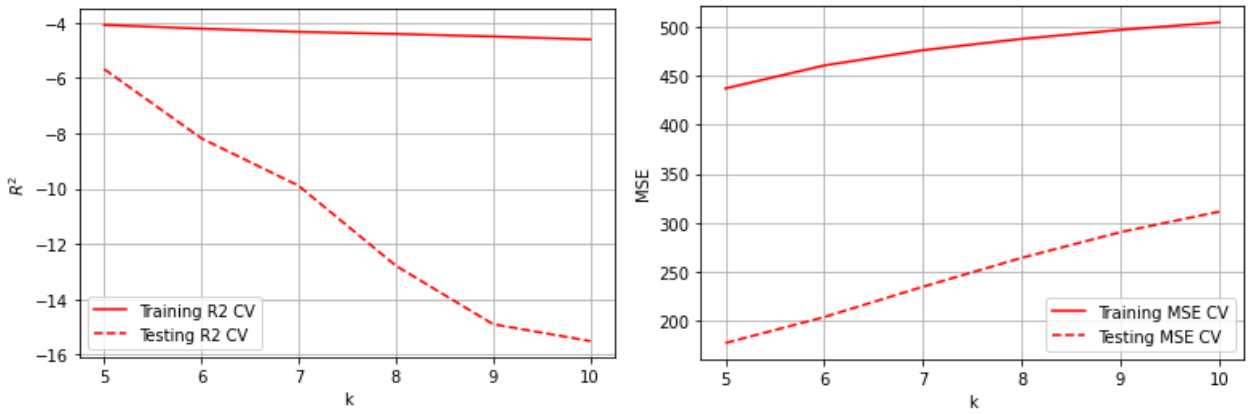


Figure 16: Chosen features



(a) $n_{feature}$=3,order=3 (b) $n_{feature}$=3,order=4 (c) $n_{feature}$=3,order=7

Figure 17: R^2, $n_{feature}$=3,order=3,4,7



(a) R^2 (b) $n_{feature}$=3,order=7

Figure 18: MSE

5 Conclusions

The project was focused on linear regression and resampling approaches. For the implementation of the analysis techniques, the Franke function and a data set of Boston Housing were provided for the project. The linear regression as ordinary least square and ridge regression were used for modeling for both data and for analysis and finding the best model values as mean of squared errors and R squared value were calculated. Moreover, the dependence of the obtained mean of squared errors and R squared values on such parameters as the standard deviation of the noise and number of data points were checked. Additionally, resampling techniques both bootstrap and cross-validation were implemented to reveal their effect on the result. In order to assess the model the bias-variance trade-off was used and the graphs were plotted to show the variance and bias vs model complexity,k-fold.

In order to perform the project the code was written, and the Scikit-learn library was applied to the data.

6 Bibliography

[1] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[2] L. Prechelt, "Early stopping-but when?," in *Neural Networks: Tricks of the trade*, pp. 55–69, Springer, 1998.

[3] J. Brownlee, "What is the difference between test and validation datasets," *Machine Learning Mastery*, vol. 14, 2017.

[4] *https://en.wikipedia.org/wiki/Biasvariance-tradeoff*.

[5] J. Mandel, "Use of the singular value decomposition in regression analysis," *The American Statistician*, vol. 36, no. 1, pp. 15–24, 1982.