UNIVERSITY OF CAEN NORMANDIE

DEPARTMENT OF PHYSICS

PROJECT 1

- report for Data analysis and machine learning course -

Professor:

Morten Hjorth-Jensen

Advisers:
Kristine B. Heine, Bendik Samseth

Students:

Igor Daskalovski, Aidana Shagalakova, Miloš Tomić, Kristina Vučković

February 2019

CONTENTS

1 ABSTRACT

Aim of this project task was to further deepen our understanding of Machine Learning and Linear Regression methods: Ordinary Least Squares (OLS), Ridge regression and Lasso regression, with polynomial fitting function of 3rd, 4th and 5th order, by implementing them in two practical examples. In the first part of the project, the focal point was Franke's function, a function that has been widely used in testing different interpolation and fitting algorithms. In the second part of the project, we have used the Python codes we have implemented in the first part and applied them to the real-life data. In our case, the dataset represented the terrain over Vojvodina, the biggest plain in Serbia.

2 INTRODUCTION

Linear regression is the process of fitting a polynomial of unknown degree to a data set. We have studied three different regression models: the ordinary least squares (OLS), Ridge and Lasso methods. We will compare these models by evaluating the mean squared error, the R^2 score function and the variance of the model parameters. We will also perform the k-fold cross validation technique.

In this section, the mathematical and computational theory of the linear regression models, necessary statistics and the resampling techniques will be defined.

2.1 LINEAR REGRESSION MODELS

Aim of the regression is to find the best fit for a data set to a model of choice. Linear models make an output that is a linear product of the predictor variables $\hat{x} = [x_0, x_1, \ldots, x_{p-1}]^T$ and the regression parameters $\hat{\beta} = [\beta_0, \beta_1, \ldots, \beta_{p-1}]$ so that:

$$\tilde{y} = \sum_{i=0}^{p-1} \beta_i x_i \tag{1}$$

The vector $\hat{x}$ is the representation of a single point in the model of choice, i.e. $\hat{x} = [1, x, x^2]^T$ for a second order polynomial. An actual data point from the data set y can be written as:

$$y = \sum_{i=0}^{p-1} \beta_i x_i + \varepsilon \tag{2}$$

where ε is the difference between the data and the model estimations:

$$y - \tilde{y} = \varepsilon \tag{3}$$

When fitting a model to a data set we get a set of equations with a goal to find a vector $\hat{\beta}$ that translates well the input predictors into the data points. Vector $\hat{\beta}$ is the same for all data point pairs (x_i, y_i), but each pair has its own predictor vector $\hat{x}$. They can be gathered in the design matrix $\hat{X}$. Set of equations for n (x, y) pairs using the elements in the design matrix can be gathered and written in matrix form:

$$\hat{y} = \hat{X}\hat{\beta} + \hat{\varepsilon} \tag{4}$$

Further expansion of the regression would be into higher dimensions, for an example the data points z as a function of both x and y. The only component that would change in this case is the design matrix $\hat{X}$ taking both variables x and y into consideration. For an example, in the second degree polynomial case, in a row in $\hat{X}$ consists of the elements $[1, x, x^2, xy, y^2]$, where each term in the polynomial is represented and consequently has its own β parameter in the $\hat{\beta}$ vector.

We can find a polynomial model that gives a (nearly) perfect fit, no matter what the underlying function in the data looks like, as long as it is continuous. When this function is disturbed by noise, altering the data set we have to work with, we do not know what part of the data is the underlying function and what is just noise. This raises a question whether our model has also fitted the noise and not the underlying function. This phenomenon is called overfitting.

There are many ways to find an estimate for the optimal $\hat{\beta}$ for a given data set (with or without noise) and a model of choice. Three of the most popular methods are the Ordinary Least Squares, Ridge and Lasso methods.

2.2 ORDINARY LEAST SQUARES

In the ordinary least squares (OLS) we are finding the β values that minimize the residual sum of squares, which is here defined as the cost function Q:

$$RSS(\beta) = Q(\beta) = \sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 = \sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1}\beta_j x_{ij}\right)^2 \tag{5}$$

or in a matrix form:

$$Q(\beta) = \left(\hat{y} - \hat{X}\hat{\beta}\right)^2 = \left(\hat{y} - \hat{X}\hat{\beta}\right)^T\left(\hat{y} - \hat{X}\hat{\beta}\right)^2 \tag{6}$$

RSS can be interpreted as the sum of all the squared errors, or distances, from the data points y to the corresponding points $\tilde{y}$ in the regressed line. To minimize the error we need to find the minimum of the cost function (here the RSS) with respect to β:

$$\frac{\partial Q(\beta)}{\partial \beta_j} = 0 \tag{7}$$

$$\frac{\partial}{\partial \beta_j}\left(\sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1}\beta_j x_{ij}\right)^2\right) = 0 \tag{8}$$

$$-2\sum_{i=0}^{n-1} x_{ij}\left(y_i - \sum_{j=0}^{p-1}\beta_j x_{ij}\right)^2 = 0 \tag{9}$$

or in a matrix form:

$$-2\hat{X}^T\left(\hat{y} - \hat{X}\hat{\beta}\right) = 0 \tag{10}$$

$$\hat{X}^T\hat{y} = \hat{X}^T\hat{X}\hat{\beta} \tag{11}$$

$$\boldsymbol{\hat{\beta} = \left(\hat{X}^T\hat{X}\right)^{-1}\hat{X}^T\hat{y}} \tag{12}$$

Differentiating again with respect to β_j we get:

$$\frac{\partial^2 Q(\beta)}{\partial {\beta_j}^2} = 2\hat{X}^T\hat{X} \tag{13}$$

where the resulting matrix is positive definite, if we assume none of the columns in $\hat{X}$ are linearly dependent. This proves that Q(β) is at a minimum in (12).

Variance of the parameters $\hat{\beta}$ tells us how sensitive the β values are to changes in the sampling, i.e. if the model was fitted on different points from the same distribution. Variance can be expressed as:

$$Var(\hat{\beta}) = \left(\hat{X}^T\hat{X}\right)^{-1}\sigma^2 \tag{14}$$

where:

$$Var(\hat{y}) = \sigma^2 = \frac{1}{N-p-1}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{15}$$

The more data points the better fit we get with this method, but if the data the model is trained on is full of noise then the OLS regression will fit to this noise. This makes the variance high, where the prediction of new data will vary greatly depending on the data it has been fitted to. Regression methods that are able to look past the noise giving better predictions of unseen data are Ridge and Lasso methods.

2.3 RIDGE METHOD

Ridge regression method modifies OLS method by adding an extra penalty term to the cost function:

$$Q(\beta) = \sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=0}^{p-1}{\beta_j}^2 \tag{16}$$

Large β values within the penalty term $\lambda|\beta^2|$ are bringing larger contributions to the cost function. Since we want to minimize the cost function, all the β values should be small, effectively penalizing the large ones and driving them down. However, punishing the β values too much sacrifices the fit given in the RSS-term, causing a low penalty but high RSS. Thus, we need to find a compromise between a good general fit (RSS) and the amplitude of the feature estimates (β values). In this way, large β values that would cause overfitting in OLS are lowered down. This process of shrinking the coefficients is called regularization, where its strength is determined by the regularization constant λ. For each model we need to test several values of λ, until we find the best fit.

Optimal regularization constant λ can be found running a cross-validation algorithm, which calculates the performance (mean squared error) for different λ values, where the λ with the lowest average error is the best one.

An expression for the optimal $\hat{\beta}$ can be derived in the similar way as for the OLS. Starting from the minimum of the cost function and modifying the following steps we end up with the expression:

$$\widehat{\boldsymbol{\beta}}^{\boldsymbol{ridge}} = \left(\widehat{\boldsymbol{X}}^{\boldsymbol{T}}\widehat{\boldsymbol{X}} + \boldsymbol{\lambda I}\right)^{-\mathbf{1}}\widehat{\boldsymbol{X}}^{\boldsymbol{T}}\widehat{\boldsymbol{y}} \tag{17}$$

where λ is in fact $\lambda \cdot I_p$ that is being added to the diagonal of the $\hat{X}^T\hat{X}$ matrix.

The variance, in this case, can be written as:

$$Var(\hat{\beta}^{ridge}) = \left(\hat{X}^T\hat{X} + \lambda I_{pp}\right)^{-1}\hat{X}^T\hat{X}\left(\left[\hat{X}^T\hat{X} + \lambda I_{pp}\right]^{-1}\right)^T\sigma^2 \tag{18}$$

where again we have:

$$\sigma^2 = \frac{1}{N-p-1}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{19}$$

The Ridge method is more a method of predicting rather than inferring. The cost of the Ridge method is always higher than the one for OLS, but the Ridge method is less likely overfit the data. This makes the model less sensitive to changes in the data set (by increasing λ, we get lower variances), at the cost of some bias. A model with a good choice of λ will generalize better the data with a lot of noise, making it better at predicting unseen points.

2.4 LASSO METHOD

The Lasso regression method is similar to the Ridge method. Its cost function can be expressed as:

$$Q(\beta) = \sum_{i=0}^{n-1}\left(y_i - \sum_{j=0}^{p-1}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=0}^{p-1}|\beta_j| \tag{20}$$

As it can be seen, the penalty term now contains absolute value $|\beta_j|$ instead of ${\beta_j}^2$ for the Ridge method. This change has a few consequences.

In contrary to the OLS and the Ridge method, in the Lasso method it is not possible to find an analytical expression for the $\hat{\beta}$ at which the cost function is minimized. That is why we have to use other minimization methods (gradient descent or Newton's method) in order to find an approximation to the minimum. To initialize the minimization we have to choose a random $\hat{\beta}$ vector. Distance from the minimum defines the number of iterations necessary for the algorithm to converge. Sometimes, it will not converge at all. Because of that, we can use Scikit learn package in Python, which will solve the minimization problem numerically, rather than analytically.

Another difference between the Lasso and the Ridge method is that the Ridge drives the β values towards, but not exactly 0, while the Lasso sets some parameters to 0, leaving only the more significant β values in the model (feature selection). This simplifies the model and also avoids overfitting. λ

parameter defines the strength of the penalization: stronger penalization lowers the variance (at the cost of some bias).

2.5 SINGULAR VALUE DECOMPOSITION

Singular value decomposition (SVD) is a method of decomposition any matrix into a product of three new matrices:

$$\hat{A} = \hat{U}\hat{D}\hat{V}^T, \tag{21}$$

where the matrix $\hat{D}$ is a diagonal matrix containing the singular values 1 of the matrix $\hat{A}$, while $\hat{U}$ and $\hat{V}$ are orthonormal matrices consisting of left-singular and right-singular vectors respectively. If the matrix $\hat{A}$ has dimensions m × n, the dimensions for $\hat{U}$, $\hat{D}$ and $\hat{V}$ will be m × m, n × n, and n × n respectively.

A problem occurs if the design matrix is nearly or fully singular (D ~ 0). This is especially challenging when the data is high-dimensional (the number of regressors β is close to or larger than the number of data points). The problem originates from the fact that a singular matrix (D = 0) has no inverse, and in order to find the regression parameters $\hat{\beta}$, we need to find the inverse of the matrix $(\hat{X}^T\hat{X})$. Using SVD we can circumvent the unstable inversion by decomposing our design matrix as described above. Such decomposition is a safe choice regardless of what our design matrix looks like.

After we apply SVD on the OLS method, we can rewrite expression (12) as:

$$\hat{\beta} = \hat{V}\hat{D}^{-1}\hat{U}^T\hat{y} \tag{22}$$

This result is called the pseudo-inverse of $\hat{X}$, where the inversion is done by calculating the inverse of the diagonal matrix $\hat{D}$, which consists of the reciprocals of the diagonal elements. In a similar way equation (14) can be expressed as:

$$Var(\hat{\beta}) = \hat{V}\hat{D}^{-2}\hat{V}^T\sigma^2 \tag{23}$$

It's also possible to derive expressions for $\hat{\beta}^{ridge}$ and $Var(\hat{\beta}^{ridge})$, but it's not strictly necessary to decompose $\hat{X}$ when using this regression method because the addition of λ in the diagonal of $\hat{X}^T\hat{X}$ vanishes the singular matrix problem. We will stick with the already derived equations for the Ridge method.

3 MODEL ASSESMENT

Properties of the model telling us something about its quality are: the overall error of the estimates (**mean squared error**), the models tendency to overestimate or underestimate (**bias**), how prone it is to change in the data set (**variance**), etc.

Another important technique for evaluation of any model is to split the data set into two parts: a training set and a test set. The training set is the data that we fit our model with. We would expect a good fit on this set as long as the model is complex enough. However, the test set consists of data points that the trained model has never seen before. The fit we get on the test set does not really tell us that much other than giving an estimate of the complexity needed to get a reasonable fit. Both sets are sampled from the same population, so a good model should perform well on both sets.

3.1 ERROR MEASUREMENTS

The means square error (MSE) defines directly how different are estimates of the model from the original data, point by point. We express MSE as:

$$MSE(\hat{y},\tilde{\hat{y}}) = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - \tilde{y}_i)^2 \tag{24}$$

This equation is similar to the RSS used in the cost function of the OLS regression method, but here the MSE calculates the mean value of the errors. The RSS increases its value with increasing number of data points, while the MSE disregards this dependence, giving a nice comparable value for data sets of different size.

When the data is split into two parts, the difference between estimations and data (MSE) on the training set goes towards 0 as complexity increases. However, on the test set we are likely to see a high MSE for both too low and too high complexities, with the minimum somewhere in the middle. It's not guaranteed that the MSE at this point is 0, due to noise in the fitted data.

The **R^2-score** is another measurement of the error, and it is defined as:

$$R^2(\hat{y},\tilde{\hat{y}}) = 1 - \frac{\sum_{i=0}^{N-1}(y_i-\tilde{y}_i)^2}{\sum_{i=0}^{N-1}(y_i-\bar{y})^2} = 1 - \frac{RSS}{TSS}, \tag{25}$$

where RSS is the residual sum of squares, TSS is the total sum of squares which is proportional to the variance of $\hat{y}$ and $\bar{y}$ is the mean value:

$$\bar{y} = \frac{1}{N}\sum_{i=0}^{N-1} y_i \tag{26}$$

The R^2-score is the coefficient of determination and describes how much of the variance in the data is accounted for by the model. This measures the likelihood of how well a model predicts future points. The RSS of the predictions is compared to model of 0^{th} order (constant) set at the average of the data points $\bar{y}$. An R^2-score equal to 1 is the best possible value and says that the predictions fit the data

perfectly. A negative R^2-score is also possible, as a fit (especially on a test set) can be worse than the 0^{th} order fit.

Additionally, the R^2-score is scale invariant, while the MSE is not. If the data set contains rather large values that are not normalized before use, the MSE-values will be large, since the squared differences may become enormous. This may cause confusion in whether the measured MSE is actually good or not. The R^2-score will always be near 1 if the model is good and negative if the fit is bad.

3.2 BIAS-VARIANCE TRADE-OFF

Let us assume that we have a set of data $y = f(x) + \varepsilon$, where $f(x)$ is the underlying function where the data comes from and ε is the random noise, which is that has the normal distribution with $E(\varepsilon) = 0$ and $Var(\varepsilon) = \sigma^2$. Thus, all randomness in y comes from ε. Because f is deterministic and $\varepsilon \in N(0, \sigma)$ we have the following implications:

$$E(\varepsilon) = 0 \tag{27}$$

$$E(f) = f \tag{28}$$

$$E(y) = E(f + \varepsilon) = f + 0 = f \tag{29}$$

$$Var(\varepsilon) = \sigma^2 \tag{30}$$

$$Var(f) = 0 \tag{31}$$

$$Var(y) = Var(f + \varepsilon) = Var(\varepsilon) = \sigma^2 \tag{32}$$

The expected value of a squared random variable is:

$$E(X^2) = Var(X) + (E(X))^2 \tag{33}$$

We define another variable $\hat{f}$ which is our model's predicted value of y. The MSE of the data points y and the predicted values $\hat{f}$ can be defined as:

$$\mathrm{E}\left(\left(\mathrm{y} - \hat{f}\right)^2\right) = \sigma^2 + Var\left(\hat{f}\right) + Bias(\hat{f})^2 \tag{34}$$

We rewrite bias in component form as:

$$Bias = \frac{1}{N}\sum_{i=0}^{N}\left(\hat{f}_i - y_i\right) = \frac{1}{N}\sum_{i=0}^{N}\left(\hat{f}_i - (f_i + \varepsilon)\right) \tag{35}$$

Bias of a model gives a measure of the tendency to overestimate or underestimate its estimates $\hat{f}$, corresponding to a positive or negative bias respectively. Usually, we have no idea what the underlying function f(x) is, all we have are the data points y with some amount of noise, but the bias is not affected by whether we know f(x) or not:

$$Bias = E(\hat{f}_i) - E(f_i) - E(\varepsilon) \tag{36}$$

Since we are dealing with a finite-sized data set, the quantity $E(\varepsilon)$ will not be exactly 0. The noise will have some impact on the bias, although its effect is small.

The MSE can be rewritten into one bias-term and two variance-terms. In practice, we can only compute these two distinct variances if we know f(x) and then extract the noise and its variance σ^2. If we only have the data points, we can only calculate the variance of our estimates $\hat{f}$ with respect to y. This makes it difficult to find a model that has both low bias and low variance, as we cannot be sure what fluctuations of the data are noise and what are the features of the underlying distributions.

That is why we have the trade-off between the bias and the variance of the model, where the goal is to find the optimal model where both quantities are as low as possible. With higher complexity of the model, the bias will drop but the variance will increase (Figure 1). Low complexity models are not able to pick up on the features in the data set, instead they tend to do underfitting, which is characterized by a high bias, but low variance. Highly complex models fit more of the fluctuations in the data, also those caused by noise. This gives a good fit but it is giving huge errors when exposed to data it has never seen before. This is called **overfitting** and it is characterized by low bias and high variance.

3.3 CONFIDENCE INTERVALS

The sample mean does not provide information about the precision of our estimation, and in general, the sample statistics is not equal to the population parameters, due to the sampling variability. Instead, we can give a confidence interval which might contain the true parameter. To calculate the confidence interval we need to select a confidence level - standard value is usually 95 %. A confidence level of 95% implies that 95% of all samples would yield an interval which includes the true parameter.

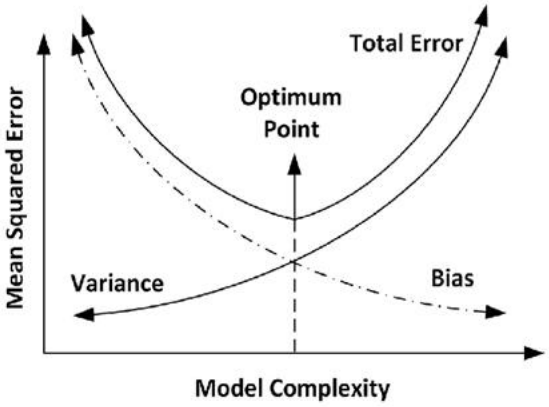

Figure 1: Bias-Variance trade-off (1)

As long as the population is normally distributed, the expectation value of the sample distribution of the sample mean $\hat{X}$ will be equal to the population mean:

$$E(\hat{X}) = \mu_{\hat{X}} = \mu \tag{37}$$

However, when the population is not normally distributed we need the central limit theorem, which states that even if the population is not normally distributed, the sample mean will be normally distributed, as long as n is large. As $n \to \infty$ the mean of this normal distribution tends towards the population mean $\mu_{\hat{X}} = \mu$, and the variance towards the population variance divided by the sample size:

$$\sigma_{\hat{X}}{}^2 = \sigma^2/n \tag{38}$$

When a random variable X is normally distributed, we compute probabilities of X by first standardized random variable:

$$Z = (X - \mu)/\sigma, \tag{39}$$

where μ is the population mean and σ is the population standard deviation. Usually, we do not know the true value of σ, so we use the sample variance $\sigma/\sqrt{n}$ as an approximation. Subtracting μ shifts the mean to 0, and dividing by σ scales the variable to get a standard deviation of 1. This gives us a relationship between the random variable X and the standard normal distribution, so we can use a table over standard normal curve areas and from this calculate the confidence interval. From the table we find that the area between -1.96 and 1.96 of the standard normal distribution is 0.95, so the relationship is:

$$P(-z < Z < z) = 0.95 \tag{40}$$

$$P(-1.96 < Z < 1.96) = 0.95 \tag{41}$$

$$P\left(-1.96 < \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} < 1.96\right) = 0.95 \tag{42}$$

$$P\left(\bar{X} - \frac{1.96\,\sigma}{\sqrt{n}} < \mu < \bar{X} + \frac{1.96\,\sigma}{\sqrt{n}}\right) = 0.95 \tag{43}$$

Equation (41) can be used to calculate the confidence interval directly. Its meaning is that we are 95% confident that the interval will contain μ. As soon as we have calculated the interval, the parameter is either in or out of the interval. However, if the experiment is repeated an infinite number of times, we would calculate an interval which contains the parameter 95% of the time.

3.4 RESAMPLING METHODS

Resampling methods draw samples from a training set and refit a model on each sample in order to obtain new information about the model, which would not be available when only fitted once to the original data.

3.4.1 k-fold cross validation

There are several hyper-parameters which have to be set before the learning or regression process can begin, for an example: the degree of our polynomial fit or the λ parameter in Ridge and Lasso regression. We need to make an educated guess for the values of these hyper-parameters, which will generate a model that generalize well to new data. First, we can split our data into training data and validation data and train a range of different models with different hyper-parameters to choose the model that fits best to the validation data. However, we have to have enough data to split it and still be confident that those parts are a good representation of our data. In many cases, the amount of data is not enough and we try a similar approach but: the k-fold cross validation.

The main steps in the k-fold cross-validation algorithm are:

- We split the training data into k equally sized parts (folds).
- We set aside one of the folds to be the validation set, and train our model for a given set of hyper-parameters on the remaining k-1 folds, then we test our model on the validation set.
- We repeat this procedure until all the folds have been validated.
- We then average the k validation-errors and this is our total estimate for the validation error.

This can be repeated for several sets of hyper-parameters and we choose the model which gives the best average of the validation-errors. In practice we are overfitting our model to the validation set, and the error from the validation will most likely be an overestimate of the true test error.

4 PROJECT TASKS

4.1 FRANKE'S FUNCTION

First we will generate some data from the Franke's function, pick out random points in the xy-plane and then calculate the corresponding function values f(x, y). Franke's function consists of several exponential functions which create two Gaussian peaks of different heights:

$$F(x,y) = \frac{3}{4}e^{-\frac{(9x-2)^2}{4}-\frac{(9y-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+1)^2}{49}-\frac{(9y+1)^2}{10}} + \frac{1}{2}e^{-\frac{(9x-7)^2}{4}-\frac{(9y-3)^2}{4}} - \frac{1}{5}e^{-(9x-4)^2-(9y-7)^2} \quad (44)$$

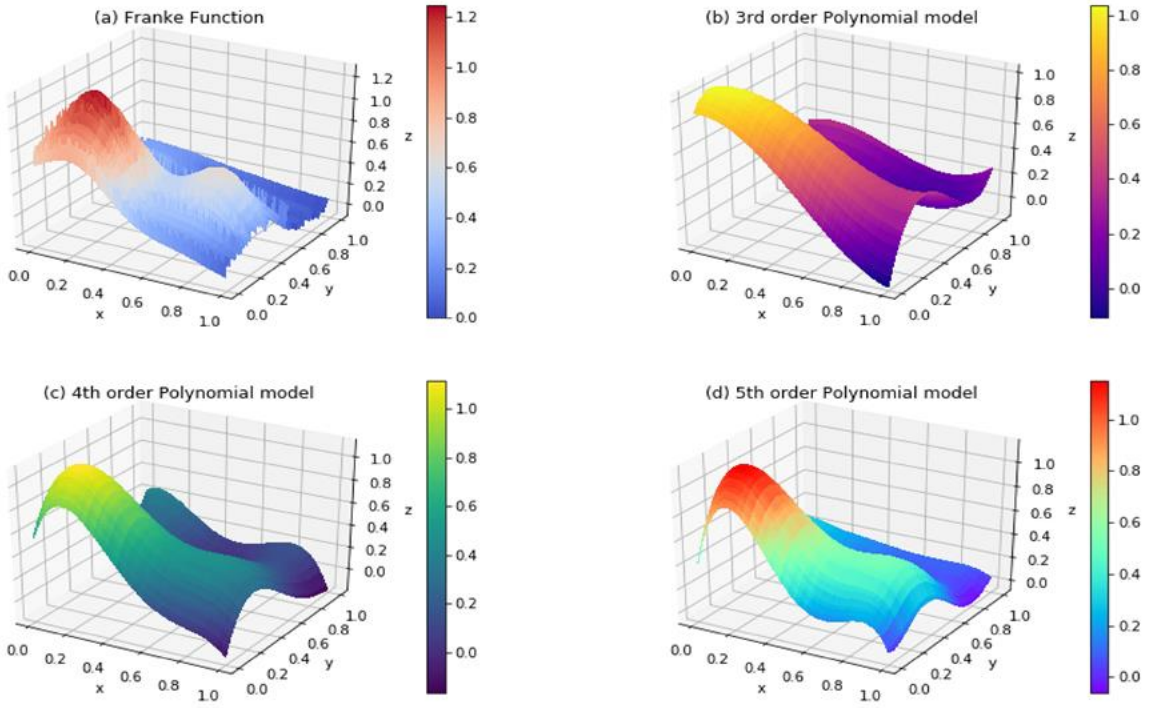

Figure 2: (a) Franke's function (b) OLS 3rd order polynomial fit (c) OLS 4th order polynomial fit (d) OLS 5th order polynomial fit

A plot of the function in the region we're interested in can be seen in Figure 2(a). Although the surface of the function is complex, it is possible to find a good fit, by approximating it with polynomials of a different degree. Figure 2 (b-d) shows the fits performed using OLS for 3rd, 4th and 5th order polynomials. Validity of these fits will be discussed in the next section.

4.2 TERRAIN DATA

In nature, the fluctuations in terrain are random due to erosion, local variations in weather, manmade structures etc. Making prediction of large and irregular area is almost impossible with simple linear regression models.

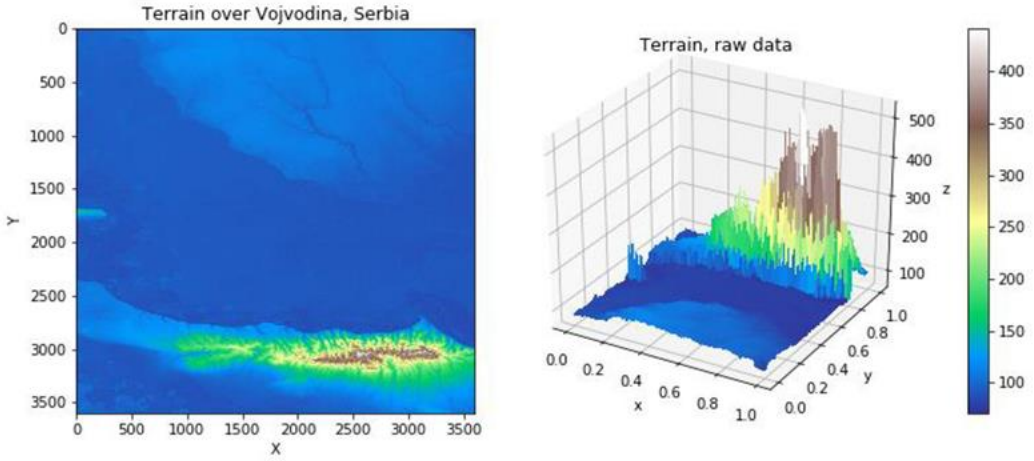

Figure 3: (a) Actual terrain over Vojvodina, largest plain in Serbia (b) Raw data plot

In Figure 3 we can see the terrain we are trying to fit with a polynomial function. We have decided to do the fitting over Vojvodina, the largest plain in Serbia. Figure 2(a) represents the actual data (3601 x 3601 pixels), but in order to run our code in a decent time, we used every 10^{th} point and in that case we reduced the size of our data set from the order of 10^6 to the order of 10^4. The raw data can be seen in Figure 2(b). The black valleys in the figure might look uniform, but there are a lot of variations in height which are not visible. So, we need an algorithm with noise regularization. A higher degree polynomials will naturally start fitting more and more irregularities.

Because of the random nature of our data the best model would be the one which would find the best compromise between fitting smallest and largest fluctuations in the dataset. As we still want to fit the biggest trends in the data, we need to regularize and penalize the higher order polynomials, hence we can expect Ridge and Lasso to make the best fits.

5 RESULTS AND DISCUSSION

5.1 FRANKE'S FUNCTION

When we analysed Franke's function, we applied 3 methods: OLS, Rigde regression and Lasso regression. All 3 methods used 3rd, 4th and 5th order polynomials as fitting function, while in Ridge and Lasso methods we introduced different values of regularization parameter λ. Validity of the obtained results was examined using R^2 score and Mean Squared Error (MSE).

5.1.1 5.1.1 OLS without resampling

Figure 2(b-d) shows the fitting results for OLS method without data resampling. As expected, as we increase the order of the polynomial, we get the better fit, which is also confirmed with R^2 and MSE values. Namely, 5th order polynomial fitting function has the lowest MSE and the highest R^2 score.

Table 1: MSE and R^2 score for OLS without resampling

Order of polynomial	Mean Squared Error	R^2 score
3	0.01615725629008327	0.8580513641976408
4	0.014726428474707304	0.8706218187856112
5	0.01302418058428749	0.8855768186633343

In this case, we also calculated variance and standard deviation for fitting parameters β, in order to obtain confidence intervals. We know from the Central Limit Theorem that 95% of our data points will be fitted if our β's fall in the [-2σ, +2σ] confidence interval, where σ represents the standard deviation of β. Because of the large number of β parameters (10 in 3rd order polynomial fitting, 15 in 4th order polynomial fitting and 31 in 5th order polynomial fitting), we omit the values of σ-s in this report, but they can still be found in the output of our Python code.

5.1.2 5.1.2 OLS with resampling

In order to improve our results, second step was to use the resampling technique on our dataset. We decided to split the data into train set and test set by using k-fold cross validation method. After resampling we repeated the OLS regression. Results for MSE and R^2 score can be found in Table 2 and Table 3. If we compare these results with the ones in Table 1, we obviously see the improvement of the fitting process.

Table 2: MSE for resampled data

Order of polynomial	Train data	Test data
3	0.011548	0.011003
4	0.006854	0.005584
5	0.004846	0.004070

Table 3: R^2 score for resampled data

Order of polynomial	Train data	Test data
3	0.889596	0.908368
4	0.933452	0.953499
5	0.952398	0.966108

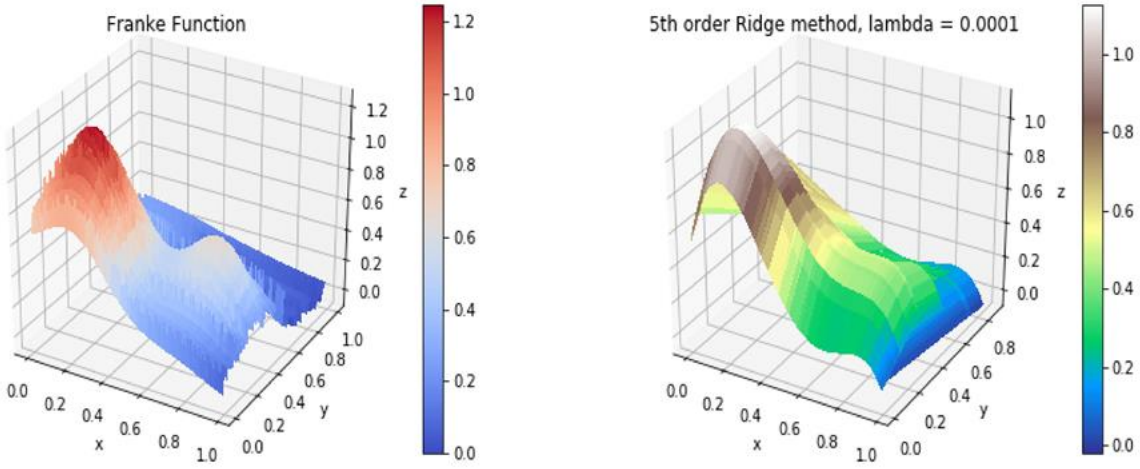

Figure 4: Best fitting for Franke's function using Ridge and Lasso method. When we compared the methods, we got the best results (R^2= 0.966230, MSE= 0.004055) for Ridge method with λ=0.0001 and 5th order polynomial fitting function

5.1.3 Ridge/Lasso method

As we have seen in sections 2.3 and 2.4, Ridge method and Lasso method are fairly similar, but the slight differences in the method can lead to significant changes in the final result. In order to compare these methods, we used the same values of penalization parameter: $\lambda \in$ (0.0001, 0.001, 0.1, 10, 100, 1000).

After we performed the calculations, based on R^2 score and MSE, we have seen that the best fit is provided by 5th order polynomial fitting function in Ridge method, for λ=0.0001. For clarity, we omit in the report tables with MSE and R^2 score. These results can be seen in the output of our Python code.

5.2 TERRAIN DATA

We performed OLS, Ridge and Lasso regression for the terrain data we introduced in section 4.2. We again used 3rd, 4th and 5th order polynomial fitting functions. But, as previously stated, we need a higher order polynomial to get a better fit, so we also used 15th order polynomial fitting function.

In Figure 5, we can see the results of OLS regression. R^2 score and MSE confirm that highest order polynomial gives the best fit.

Table 4: R^2 score and MSE of OLS regression for terrain data

Order of polynomial	Mean Squared Error	R^2 score
3	1588.948911129354	0.2585449270320268
4	1307.0572940858717	0.39008469399374746
5	1106.2244002828954	0.4837998329048012
15	354.46917971563835	0.8345931894536789

We have also applied Ridge and Lasso methods onto our terrain data, and we got the best fit with 15th order polynomial as a fitting function, for λ=0.0001 (MSE= 561.791261, R^2= 0.705357).

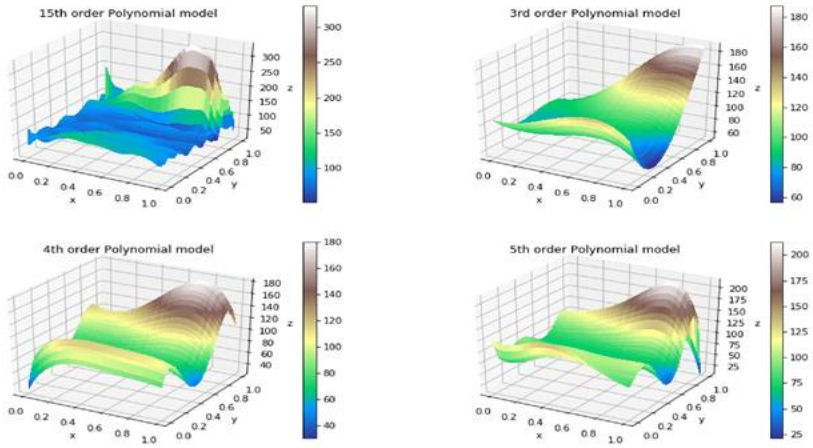

Figure 5: OLS regression with polynomial fitting functions of different order. If we compare them to terrain raw data on Figure 3(b), we see that the highest order polynomial gives the best fit.

Again, we do not present here MSE and R^2 score for all cases we analysed, but they can be found in the output of our Python code.

As we have shown in the previous section, among different values of λ we have selected for Ridge and Lasso methods, λ=0.0001 for Rigde method gives out the best results, which get even better as we increase the order of the polynomial that we use as our fitting function.

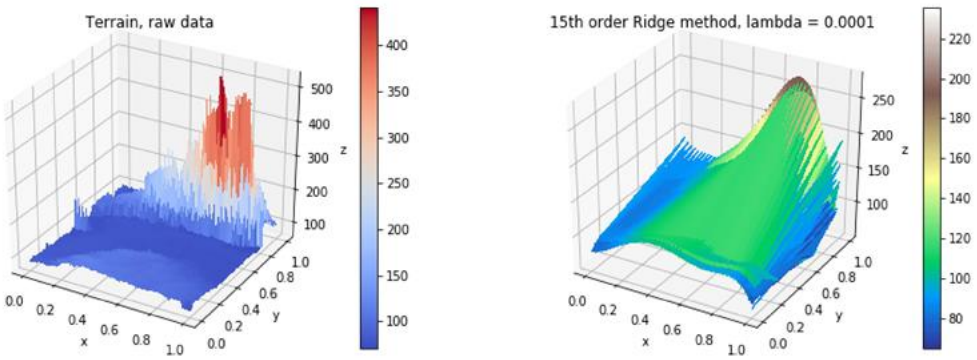

Figure 6: (a) Terrain, raw data (b) Best fit for Ridge and Lasso method (Ridge method, λ=0.0001)

6 CONCLUSION AND OUTLOOKS

Aim of this project task was to further deepen our understanding of Machine Learning and Linear Regression methods, by implementing them in two practical examples. In the first part of the project, the focal point was Franke's function, a function that has been widely used in testing different interpolation and fitting algorithms. We have used three regression methods: Ordinary Least Squares (OLS), Ridge regression and Lasso regression, with polynomial fitting function of 3^{rd}, 4^{th} and 5^{th} order. Also, in order to get a better assessment of our model, we have employed the k-fold cross-validation resampling technique on our dataset.

For Ridge regression and Lasso regression we have also used the regularization parameter (as described in sections 2.3 and 2.4) which was selected from the set of fixed values, $\lambda \epsilon$ (0.0001, 0.001, 0.1, 10, 100, 1000).

Our main parameters in assessing the quality of our model were Mean Squared Error (MSE) and R^2 score. The model is better if MSE is closer to 0 and R^2 score is as close as possible to 1. In the case of OLS, we have also found the confidence interval, by using the standard deviation and Central Limit Theorem.

As it can be seen in section 5.1, the quality of our model increases as we increase the order of our polynomial fitting function. In section 3.2 it is shown that increasing complexity of the model will eventually lead to worse results (see Figure 1), but we have not seen that kind of a behaviour in our model, because we have only used lower order polynomials as fitting functions. More complex models with higher order polynomials should be investigated in the future.

In case of the Ridge and Lasso regression methods, we have obtained the best results for λ=0.0001, which suggests that we have to carefully select this parameter. λ has to be substantially small (or large) depending on other specific features of the model.

In the second part of the project, we have used the Python codes we have implemented in the first part and applied them to the real-life data. In our case, the dataset represented the terrain over Vojvodina, the biggest plain in Serbia. It was clear in the very beginning that 3^{rd}, 4^{th} or 5^{th} order polynomial as a fitting function will give out a really bad results, so we decided to include a 15^{th} order polynomial as well. Naturally, it gave out the best results, both in OLS and in Ridge/Lasso method. Even higher order polynomial would possibly give out even better results, but that would increase the running time of the code. Finding an optimal solution for this problem (better fitting, but appropriate running time) is also one of the tasks we should look into in the future.

7 BIBLIOGRAPHY

1. Detect and charge: Machine learning based fully data-driven framework for computing overweight vehicle fee for bridges. **Gungor, Osman Erman.** Illinois : Elsevier, 2018, Vol. 96. https://doi.org/10.1016/j.autcon.2018.09.007.

2. **Shalizi, Cosma Rohilla.** Advanced Data Analysis from an Elementary Point of View. 2018.

3. **Hastie, Tibshirani, Friedman.** The Elements of Statistical Learning. New York : Springer, 2016. 0172-7397.

4. **Andreas C. Mueller, Sarah Guido.** Introduction to Machine Learning with Python. s.l. : O'Reilly, 2016. 978-1-491-91721-3.

5. **Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani.** An Introduction to Statistical Learning. New York Heidelberg Dordrecht London : Springer, 2013. 1431-875X.

6. A high-bias, low-variance introduction to Machine Learning for physicists. **Pankaj Mehta, Ching-Hao Wang, Alexandre G. R. Day, Clint Richardson.** Boston : Boston University, 2018. arXiv:1803.08823v1 .

7. Lecture notes on ridge regression. **Wieringen, Wessel N. van.** Amsterdam : s.n., 2018. arXiv:1509.09169v3.

8. **Hjorth-Jensen, Morten.** Machine Learning Course Notes. [Online] January 2019. https://github.com/CompPhysics/MLErasmus.