**Study the effect of training set noise over different regression algorithms and settings**

Alevizopoulou Sofia, AM: 2022201704002; Avgeros Giannis, AM: 2022201704003

Tsiatsios Georgios, AM: 2022201704023, Lefteris Balteas AM

Athens 2018, MSC Data Science, Applied Data Science

# **Abstract**

It is very common machine learning techniques to deal with noisy data, which may affect the accuracy of the resulting data models. For that reason, effective dealing with noise is a key aspect in machine learning to obtain reliable models from data. In this assignment, we address this issue by comparing how the noise affect linear, Lasso and Ridge regression on the final outcomes of 3 different datasets. We will compare the final outcomes from the original datasets with the outcomes from the noisy datasets. [ADD RESULTS OF REGRESSORS, poso diaforetika ephreazetai o kateh algorithmos me noisy data]

## **Keywords**

Attribute noise, noisy data, performance of regression algorithms, statistical test, noise effects

# 1. Introduction

Machine learning techniques and more specifically supervised learning techniques, are applied to extract novel and interesting information from data collected out of real-world problems. An important characteristic of these datasets is that the data frequently contains noise. Noisy data may be the consequence of human error due to mistakes in the translation phase or due errors while collecting them. Noisy data, may produce bias to the learning process, and as a result, is more difficult for learning algorithms to form accurate models from them. Developing learning techniques that effectively deal with noisy data is a key aspect in machine learning.

        In this assignment we provide a comparison of the effect of attribute noise and class noise on three regression models (a) Linear Regression, (b)Lasso Regression, (c) Ridge Regression. We are going to use 3 different datasets from different domains, each of them describes a different regression problem. The first dataset is the *Boston Housing* dataset thatconcerns housing values in suburbs of Boston. [4] The second is *air Qualit*y dataset that contains instances of hourly averaged responses from metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. [5] And the third dataset, *nba players* generated by us and contains info about the performance of nba players. We used these datasets to generate the “noisy” version of them. Finally, having the clean and the noisy version of each dataset, we compared the effect of noise on the models created (9 different models, for each dataset 2 different kind of noise).

Training a linear regression model say we try to find out coefficients for the linear function that best describe the input variables by choosing a function to help us measure the error, this function called cost function. In regression problems, we need evaluation metrics designed for comparing continuous values. Mean Square Error (MSE) is the most commonly used regression loss function. MSE is the sum of squared distances between our target variable and predicted values. We would like to minimize the MSE function as much as possible, so the prediction will be as close as possible to the ground truth by updating the coefficients.

The results enable us to highlight the characteristics of the different approaches. The whole analysis is developed from the following null (initial) hypothesis.

*Initial hypothesis:* Given the characteristics of each learning technique, we initially propose that noisy training data don’t affect the final outcome of regression algorithms: Linear Regression, Ridge Regression and Lasso Regression

The structure of the present paper is as follows: in Sect. [2](#_2._Related_work) summarization related work in noise analysis. In Sect. [3](#_3._Regression_algorithms) presentation of the three Regression algorithms and the learning techniques. Sect. [4](#_4._Design_of) description of the data sets, and explanation of the methodology used for the noise generation. Sect. [5](#_.5._Results_of) presents and analyzes the results of the different experiments.  Sect.[6](#_6._Hypothesis_testing) describe and run hypothesis tests based on null hypothesis.  Sect. [7](#_7._Conclusions) summarization the overall conclusions.

# 2.Related work

        Understanding the impact of noisy data in the performance of machine learning algorithms is a key issue for improving algorithms reliability. Below will be described some techniques of noise generation. In current researches, some machine learning techniques are considered more “robust” to noise, errors and missing values than others.

[DO WE NEED MORE ??]

## 2.1.   Noise Generation

Noise generation can be described in different ways. First, it is very important where the noise is introduced. Noise can be introduced in the input attributes or/and the target feature of the data (feature because it is a regression problem). Second, the distribution that noise follows, for example, normal or Gaussian. Third, the generated noise values can be relative to min, max, std (standard deviation) of each variable or to the variable value itself. Noisy training data will impact on the outcomes of the final level.

We have implemented 1 python script which generates 2 noisy datasets – all features are noise, target feature is noise - based on an input dataset. The process is described below.

### 2.1.1. Attribute noise

This kind of noise adds noisy data at a specific feature of the train dataset. We don’t modify test data. First, we define the percentage of noise which can be between 0%-100%, let that be the Fc variable. After using this percentage, we find out how many random variables will be generated by multiplying the percentage of noise with the number of instances, so we have N random values. Now, for each variable of this attribute, we generate N random numbers Rv with a Gaussian distribution and within a range between the max and min of the corresponding variable. Then we generate another random numbers Rc with a uniform distribution within the range 1 to the number of instances, which indicates the instance whose data value should be overwritten by the generated noisy value. [1]

Specifically, we assign create a dictionary of lists. Each element of the dictionary is of the type {Feature name: list}, where each list contains N (number of instances) elements. Each element is assigned a probability, which represents the numerical probability of the record’s feature value to be altered by adding noise. We achieve by that to introduce noise per feature.

For example, if we assign a probability of 0.1% to add noise, that means that values produced out of the gaussian distribution have a probability (given by the beforementioned dictionary of lists) that must be below that threshold in order for the noise value to be added into the records feature value. Adding value per feature (and not per whole record) simulates at best the real case scenario of a non-simulated dataset.

### 2.1.2. Target feature noise

This kind of noise adds noisy data at the target feature of the train dataset. We don’t modify test data at this case also. The process is the same as before, as we seek to generate noise values for one feature instead of the multiple features we did on the previous feature case.

## 2.2 Stratified split for test train dataset

We wanted to create two different datasets for the implementation of noise effect. The first dataset would be the train one and the second one would be the test one. The test dataset will not contain any noise as stated previously. For the train dataset, we create two different versions of it, one with noise for the attribute and one with noise for the class dataset.

What is important here is the fact that we needed to perform the split in a stratified way. Thus, we wanted for the test and the train distribution to be as similar as possible. Due to the fact that for regression purposes, sciikit learning was not able to perform the split, we decided to make it manually, implementing it as explained below:

1. We sort the dataset based on the target feature
2. We split it the sorted dataset to lists of 10 items.
3. The proportion of split to test and train is the number of items we select out of the 10 items. We decided to split the dataset based on 80 % train – 20% test, so the number of elements taken from the sorted 10 items would be 2 items. Those 2 items would be added to the test dataset, whereas the remaining 8 are part of the training dataset.

As we see by that, we force with the way we create the datasets that the distribution of the train dataset would be preserved and be present also to the test dataset.

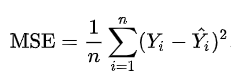
# 3. Regression algorithms

## 3.1    Regression techniques

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent and independent variable. This technique is used for forecasting, time series modelling and finding the curve / line to the data points, in such a manner that the differences between the distances of data points from the curve or line is minimized.

        One of the first steps in a regression analysis is to determine if multicollinearity is a problem. Multicollinearity, or collinearity, is the existence of near-linear relationships among the independent variables. Multicollinearity causes two basic types of problems. The coefficient estimates can swing rapidly based on which other independent variables are in the model, so coefficients become very sensitive to small changes in the model. Multicollinearity reduces the precision of the estimate coefficients, which makes the statistical power of the regression model weak.

In regression problems, we need evaluation metrics designed for comparing continuous values. We would like to minimize the MSE (loss function) for a specific data-point as much as possible so the prediction will be as close as possible to the ground truth. This is done by learning the parameters, executing an iterative process that updates coefficients at every step and reduce the loss function as much as possible until the minimum point of the loss function has been reached.



Where n: number of instances, Y^: predicted values, Y: observed values

## 3.2.   Linear regression

Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line. The dependent variable is continuous, independent variable(s) can be [continuous or discrete](https://en.wikipedia.org/wiki/Continuous_and_discrete_variables), and the regression line is linear.

Its equation is *Y = a+b1X1+b2X2+b3X3+…*

where Y: the response, a: the intercept, b: model coefficients, Xn (the nth feature)

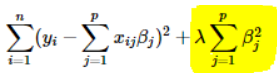
This equation can be used to predict the value of target variable based on given predictor variable(s). We would like to minimize the MSE (loss function) for a specific data-point as much as possible. Linear Regression is very sensitive to Outliers. Having noisy data can influence on the computation of MSE and as a result the regression line. Prediction errors can occur due to two sub components or combination of them. First component is the bias and the second is the variance (multicollinearity problem). [2]

## 3.3.   Ridge Regression

Ridge regression adds “squared magnitude” of coefficient as penalty term to the loss function, it is known as L2 regularization. It is another regression method and used when the data suffers from multicollinearity, independent variables are highly correlated. In multicollinearity, even though the least squares estimates are unbiased, their variances are large. Large variances mean that the observed value is far from the true value. Adding a degree of bias, standard errors are reduced. Ridge regression solves the multicollinearity problem through [shrinkage parameter](https://en.wikipedia.org/wiki/Shrinkage_estimator) λ (lambda). Ridge regression is a well-known approach to dealing with noisy data. [4]

Its equation is Y=a+b\*X+ε

where Y: the response, a: the intercept, b: model coefficients, Xn (the nth feature), ε: the error. Error term is the value needed to correct the prediction error between the observed and predicted value. The highlighted part below represents L2 regularization element. [6]



If *lambda* is very large then it will add too much weight and it will lead to under-fitting, if *lambda* equals to zero then we have linear regression. We can understand that it’s important how *lambda* is chosen. Minimizing the sum of the squares of the vertical deviations from each data point to the line enforce β coefficients to be lower but it does not enforce them to be zero.

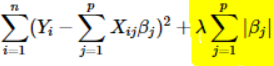
## 3.4.   Lasso Regression

Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “*absolute value of magnitude*” of coefficient as penalty term to the loss function, its known as L1 regularization. It penalizes the absolute size of the regression coefficients, reduces the variability and improving the accuracy of linear regression models.

Lasso regression differs from ridge regression in a way that it uses absolute values in the penalty function, instead of squares. This leads to penalizing values which causes some of the parameter estimates to turn out exactly zero. Larger the penalty applied, further the estimates get shrunk towards absolute zero. Therefore, you might end up with fewer features included in the model than you started with, which is a huge advantage. Lasso regressor is not equipped to deal with noisy or missing data. [2]

Its equation is Y=a+b\*X+ε

where Y: the response, a: the intercept, b: model coefficients, Xn (the nth feature), ε: the error. Error term is the value needed to correct the prediction error between the observed and predicted value. The highlighted part below represents L1 regularization element. [6]



If *lambda* is a large value will make coefficients zero hence it will under-fit, if *lambda* equals to zero then we have linear regression. The key difference between Lasso and Ridge techniques is that Lasso shrinks the less important feature’s coefficient to zero thus, removing some feature altogether. So, this is effective for feature selection in case we have a very big number of features.

# 4. Design of the dataset

In our experiments we will use the 3 different datasets. The two of them: boston-houses, air-quality are available on web and the third is generated by us: nba-players. The nba-players dataset designed to have no noise. The other datasets as they are based on real data (data descriptions of real objects or events), may contain a certain amount of background noise, erroneous values, and so on. All the data sets have as target numerical attributes. Forced noise will be added only at the train data. Before adding noise, we have splitted the dataset into train and test.

## 4.1. Nba-players

The nba-players dataset contains information about the statistics of nba-players and made by us. More specifically each feature is a different metric of performance. GFor example 3-point, 2-ponit etc. The information has been selected from web. The target feature is the salary of the player. At this case, we try to find the best regression model at which the predict of salary will be predicted.  There is a readme file available at which there are detailed information about the dataset. There is a readme file available at which there are detailed information about the dataset. [9]

As said before, this is a dataset generated by us there are no noisy data. We generated other datasets based on this at which we will include several noise levels. Some generated noisy datasets will contain noise data at the target class - SALARY - whereas other contain noise data at all features of the dataset except the target one. The generation of noise has been analyzed on Sect. [2.1.](#_2.1.__)

## 4.2. Boston houses

The Boston dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. The dataset contains a total of 506 instances and 14 attributes. The 14 features give several information for the specific area, eg: the feature CRIM describes the crime rate by town. At this case, we try to find the best regression model at which the median value of houses will be predicted.  There is a readme file available at which there are detailed information about the dataset. [7]

As said before, this is a real dataset so for sure it contains noisy data. Because we don’t know exactly the percentage of noise we generated some other datasets based on this at which we will include several noise levels. The generated datasets contain noise data at the target feature -MEDV (median value of houses) or at all features of the dataset except the target one. For example, if the dataset contains 100 instances and 3 features (one of them is the target), 15% noise on the target feature will overwrite target value on 15 instances, whereas 15% noise on features in general will changes 15 instances at all features. The generation of noise has been analyzed on Sect. [2.1.](#_2.1.__)

## 4.3. Air-Quality

The air quality dataset contains 9358 instances of hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was located on the field in a significantly polluted area, at road level, within an Italian city. Data were recorded from March 2004 to February 2005 (one year) representing the longest freely available recordings of on field deployed air quality chemical sensor devices responses. At this case, we try to find the best regression model at which the predict the amount of C6H6(GT) will be predicted.  There is a readme file available at which there are detailed information about the dataset. [8]

As said before, this is a real dataset so for sure it contains noisy data. Because we don’t know exactly the percentage of noise we generated other datasets based on this at which we will include several noise levels on target feature or at all features of the dataset. The first generated dataset contains noise data at the target class - C6H6(GT) - whereas the other contains noise data at all features of the dataset except the target one. The generation of noise has been analyzed on Sect. [2.1.](#_2.1.__)

# Results of the experiments

At all cases we haven’t added noise on test data. Test dataset is the same for all different cases. We have run all regression problems for 6 different noisy data.

0.xx\_f\_train\_noisy.xlsx🡪 x% noise at all the features of the dataset

0.xx\_c\_train\_noisy.xlsx🡪 x% noise at the target feature of the dataset

## Nba-players

Regarding the nba players regression problem first, we found out the best model by tunning the hyperparameters and after use this model to make prediction on the test data. The result we got running the 3 regression algorithms for different noise levels and types of noise are listed below. We have used the root mean squared error as performance measure. It is obvious that using noisy data to train the model we result in bigger RMSE, so the “fit” of the regression line is getting worst.

Linear

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 3907600.1591 |
| 0.05\_c\_train\_noisy.xlsx | 3992857.6040 |
| 0.05\_f\_train\_noisy.xlsx | 3956455.5601 |
| 0.15\_c\_train\_noisy.xlsx | 4396256.8619 |
| 015\_f\_train\_noisy.xlsx | 4092153.9255 |
| 0.35\_c\_train\_noisy.xlsx | 6156055.3937 |
| 0.35\_f\_train\_noisy.xlsx | 4207287.6076 |

Lasso

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 3881398.4215 |
| 0.05\_c\_train\_noisy.xlsx | 3981004.0542 |
| 0.05\_f\_train\_noisy.xlsx | 3956455.6850 |
| 0.15\_c\_train\_noisy.xlsx | 4205175.0322 |
| 0.15\_f\_train\_noisy.xlsx | 4092153.1844 |
| 0.35\_c\_train\_noisy.xlsx | 6113698.4837 |
| 0.35\_f\_train\_noisy.xlsx | 4207285.3736 |

Ridge

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 3934420.8139 |
| 0.05\_c\_train\_noisy.xlsx | 4023162.6035 |
| 0.05\_f\_train\_noisy.xlsx | 3952206.6649 |
| 0.15\_c\_train\_noisy.xlsx | 4259801.4752 |
| 0.15\_f\_train\_noisy.xlsx | 4078612.4491 |
| 0.35\_c\_train\_noisy.xlsx | 6023397.1469 |
| 0.35\_f\_train\_noisy.xlsx | 4183776.2731 |

Except the RMSE measurement and the variances we will also use the predicted target feature values (predicted values from the model) of the test dataset using the different models in order to calculate the statistical tests.

## Boston-housing

Regarding the boston houses regression problem first, we found out the best model by tunning the hyperparameters and after use this model to make prediction on the test data. The result we got running the 3 regression algorithms for different noise levels and types of noise are listed below. We have used the root mean squared error as the nba-players dataset We can see that using noisy data to train the model we result in bigger RMSE, so the “fit” of the regression line is getting worst.

Linear

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 5.3624 |
| 0.05\_c\_train\_noisy.xlsx | 5.5905 |
| 0.05\_f\_train\_noisy.xlsx | 5.6751 |
| 0.15\_c\_train\_noisy.xlsx | 6.2990 |
| 015\_f\_train\_noisy.xlsx | 5.9265 |
| 0.35\_c\_train\_noisy.xlsx | 9.3037 |
| 0.35\_f\_train\_noisy.xlsx | 6.6809 |

Lasso

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 7.6768 |
| 0.05\_c\_train\_noisy.xlsx | 7.9135 |
| 0.05\_f\_train\_noisy.xlsx | 8.0181 |
| 0.15\_c\_train\_noisy.xlsx | 8.6815 |
| 0.15\_f\_train\_noisy.xlsx | 8.9174 |
| 0.35\_c\_train\_noisy.xlsx | 10.9375 |
| 0.35\_f\_train\_noisy.xlsx | 9.1617 |

Ridge

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 5.6751 |
| 0.05\_c\_train\_noisy.xlsx | 5.6204 |
| 0.05\_f\_train\_noisy.xlsx | 5.6863 |
| 0.15\_c\_train\_noisy.xlsx | 6.2957 |
| 0.15\_f\_train\_noisy.xlsx | 5.9473 |
| 0.35\_c\_train\_noisy.xlsx | 9.2579 |
| 0.35\_f\_train\_noisy.xlsx | 6.6934 |

[results]

## Air-quality

Regarding the air-qulaity houses regression problem first, we found out the best model by tunning the hyperparameters and after use this model to make prediction on the test data. The result we got running the 3 regression algorithms for different noise levels and types of noise are listed below. We have used the root mean squared error as the nba-players dataset. We can see that using noisy data to train the model we result in bigger RMSE, so the “fit” of the regression line is getting worst.

Linear

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 10.8003 |
| 0.05\_c\_train\_noisy.xlsx | 10.5905 |
| 0.05\_f\_train\_noisy.xlsx | 29.5640 |
| 0.15\_c\_train\_noisy.xlsx | 11.1289 |
| 015\_f\_train\_noisy.xlsx | 44.3578 |
| 0.35\_c\_train\_noisy.xlsx | 11.5400 |
| 0.35\_f\_train\_noisy.xlsx | 64.1151 |

Lasso

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 18.1186 |
| 0.05\_c\_train\_noisy.xlsx | 18.1439 |
| 0.05\_f\_train\_noisy.xlsx | 30.5056 |
| 0.15\_c\_train\_noisy.xlsx | 18.7156 |
| 0.15\_f\_train\_noisy.xlsx | 43.2067 |
| 0.35\_c\_train\_noisy.xlsx | 18.2428 |
| 0.35\_f\_train\_noisy.xlsx | 62.8595 |

Ridge

|  |  |
| --- | --- |
| **Datasets** | **RMSE** |
| clean data: | 10.9116 |
| 0.05\_c\_train\_noisy.xlsx | 10.7243 |
| 0.05\_f\_train\_noisy.xlsx | 29.5641 |
| 0.15\_c\_train\_noisy.xlsx | 11.2397 |
| 0.15\_f\_train\_noisy.xlsx | 44.3492 |
| 0.35\_c\_train\_noisy.xlsx | 11.6337 |
| 0.35\_f\_train\_noisy.xlsx | 64.1083 |

[RMSE for feature noise has very very big RMSE ?????? ]

# Hypothesis testing

## Null hypothesis

Hypothesis testing is a way to test the results of a survey or experiment. As first step the null hypothesis should be defined.  The null hypothesis is a general statement or default position that there is not, or there is a relationship between some measured phenomena. Hypothesis testing using some statistical tests which will reject or no the null hypothesis. If the null hypothesis will be rejected then the initial hypothesis we have done is fault, otherwise is true.

*Null hypothesis – H0:* Given the characteristics of each learning technique, we initially propose that noisy training data don’t affect the final outcome of regression algorithms: Linear Regression, Ridge Regression and Lasso Regression, using 3 datasets an dteh noisy versions of them. Means that samples have the same mean.

*Alternative hypothesis – H1:* Given the characteristics of each learning technique, noisy training data affect the final outcome of regression algorithms: Linear Regression, Ridge Regression and Lasso Regression, using 3 datasets an dteh noisy versions of them.. Means that samples have different mean.

## Two sample t-test

As mentioned before, we are interested in comparing the performance of the different learning techniques on data with different proportions of attribute noise and class feature noise in the training data set, using the same test dataset. For that scope we will use the Independent 𝑡-test that is used to determine if two population means are equal or not. More specifically a paired t-test. It is about doing two different tests on same dataset (the test dataset is the same, we just change the train dataset).

With a two-sample t test, we are comparing the means for two different samples. Paired t-test is used when there are two measurements on the same item. [T](http://www.statisticshowto.com/t-score-formula/) score is a ratio between the **difference between two groups.** The larger the t score, the more difference there is between groups. Each t-value has a [p-value](http://www.statisticshowto.com/p-value/) to go with it. A p-value is the [probability](http://www.statisticshowto.com/probability-and-statistics/probability-main-index/) that the results from your sample data occurred by chance. P-values are from 0% to 100%. In most cases, a p-value of 0.05 (5%) is accepted to mean the data is valid.

As we would like to compare the noisy datasets with the clean version of them we will use this type of statistical test, two sample t-test. At all cases we will use the clean and the noisy version of a dataset and we will do a two samples t-test with the predicted target values of these datasets (test dataset) in order to reject the null hypothesis and accept the alternative hypothesis.

## Running the hypothesis testing

Using the predicted values of the target feature, the means and the variances from clean and noisy data, we calculated the t-test using the *ttest* from *spicy*.

Null hypothesis will be rejects if the calculated p-value is much bigger than the significance level, at point 0.05 then the test concludes that there is a statistically significant difference between the two populations. In any other case, there is no statistically significant difference between the two populations and the test fails to reject the null hypothesis.

### 6.3.1. Nba-players

Linear

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = -0.9778151969647924  p = 0.6591771737882466 | Rejected |
| 0.05\_f\_train\_noisy.xlsx | t = 0.6558804627836343  p = 1.0256181380395115 | Rejected |
| 0.15\_c\_train\_noisy.xlsx | t = -3.712965536697131  p = 0.0005587250244903482 | Accepted |
| 015\_f\_train\_noisy.xlsx | t = 1.056689513502943  p = 0.5843773727792039 | Rejected |
| 0.35\_c\_train\_noisy.xlsx | t = -8.96936782818747  p = 1.21822146070917e-15 | Accepted |
| 0.35\_f\_train\_noisy.xlsx | t = 1.1478963291980364  p = 0.5053258690833944 | Rejected |

Lasso

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = -1.070758795079115  p = 0.5716645247311801 | Rejected |
| 0.05\_f\_train\_noisy.xlsx | t = 0.5575398419471272  p = 1.155819399098028 | Rejected |
| 0.15\_c\_train\_noisy.xlsx | t = -3.53576916673002  p = 0.0010542075148508249 | Accepted |
| 0.15\_f\_train\_noisy.xlsx | t = 0.9600057626052587  p = 0.6768970389640026 | Rejected |
| 0.35\_c\_train\_noisy.xlsx | t = -8.48415199467026  p = 2.3021785869356512e-14 | Accepted |
| 0.35\_f\_train\_noisy.xlsx | t = 1.055834322390093  p = 0.5851562451401691 | Rejected |

Ridge

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = -1.1300986557310786  p = 0.5201298062257809 | Rejected |
| 0.05\_f\_train\_noisy.xlsx | t = 0.45137237102003114  p = 1.304620635709734 | Rejected |
| 0.15\_c\_train\_noisy.xlsx | t = -3.7187850600604038  p = 0.0005469831393795063 | Accepted |
| 0.15\_f\_train\_noisy.xlsx | t = 0.849174727211056  p = 0.7940140609220696 | Rejected |
| 0.35\_c\_train\_noisy.xlsx | t = -8.448391606015514  p = 2.852368203027187e-14 | Accepted |
| 0.35\_f\_train\_noisy.xlsx | t = 0.946760256505377  p = 0.690274180437144 | Rejected |

At all the above case, running the noisy datasets we can see that p-value is much bigger than 0.05 which means that we can reject null hypothesis for the nba-palyers dataset and we can conclude that adding noisy data on this regression problem will affect the final outcomes of the model.

### 6.3.2. Boston-housing

Linear

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = -1.28788315192938  p = 0.3985201217927078 | Accepted |
| 0.05\_f\_train\_noisy.xlsx | t = -0.09860191921658555  p = 1.843104244415307 | Rejected |
| 0.15\_c\_train\_noisy.xlsx | t = -3.5635178403836836  p = 0.0009131304007021252 | Accepted |
| 015\_f\_train\_noisy.xlsx | t = -0.2611280177769364  p = 1.588519641605346 | Rejected |
| 0.35\_c\_train\_noisy.xlsx | t = -7.744200419411042  p = 9.112718431704457e-13 | Accepted |
| 0.35\_f\_train\_noisy.xlsx | t = -0.4113542578949713  p = 1.362497774385988 | Rejected |

Lasso

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = -3.6783706672850025  p = 0.0006019454602408097 | Accepted |
| 0.05\_f\_train\_noisy.xlsx | t = -0.060276784349073236  p = 1.9039895690320545 | Rejected |
| 0.15\_c\_train\_noisy.xlsx | t = -10.293008911167354  p = 6.053896322497681e-20 | Accepted |
| 0.15\_f\_train\_noisy.xlsx | t = -0.3312091914139473  p = 1.4816596304589353 | Rejected |
| 0.35\_c\_train\_noisy.xlsx | t = -22.31466636419465  p = 4.091789998345941e-56 | Accepted |
| 0.35\_f\_train\_noisy.xlsx | t = -0.39189052506939315  p = 1.3911044653815594 | Rejected |

Ridge

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = -1.3122976670638626  p = 0.38181783439366773 | Accepted |
| 0.05\_f\_train\_noisy.xlsx | t = -0.09444386145906118  p = 1.8497003821424205 | Rejected |
| 0.15\_c\_train\_noisy.xlsx | t = -3.646660134807599  p = 0.0006760243587987187 | Accepted |
| 0.15\_f\_train\_noisy.xlsx | t = -0.2584428950125643  p = 1.5926565086487434 | Rejected |
| 0.35\_c\_train\_noisy.xlsx | t = -7.985057488126355  p = 2.0896321402789618e-13 | Accepted |
| 0.35\_f\_train\_noisy.xlsx | t = -0.4168450568370551  p = 1.3544687439684948 | Rejected |

At most of the above cases, running the noisy datasets we can see that p-value is much bigger than 0.05 which means that we can reject null hypothesis for the boston-housing dataset and we can conclude that adding noisy data on this regression problem will affect the final outcomes of the model. The majority of cases reject the null hypothesis, so we can reject it in general.

### 6.3.3. Air-quality

Linear

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = 0.7639511380880694  p = 0.8898890538277886 | Rejected |
| 0.05\_f\_train\_noisy.xlsx | t = -14.191426535080941  p = 2.9088668450973627e-44 | Accepted |
| 0.15\_c\_train\_noisy.xlsx | t = 0.6515685194721174  p = 1.0294391492038142 | Rejected |
| 015\_f\_train\_noisy.xlsx | t = -26.9442889504523  p = 5.330809031997706e-146 | Accepted |
| 0.35\_c\_train\_noisy.xlsx | t = 1.1204862550279004  p = 0.5251570257694668 | Rejected |
| 0.35\_f\_train\_noisy.xlsx | t = -47.77372032772085  p = 0.0 | Accepted |

Lasso

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = 0.7593586903919249  p = 0.8953716989328284 | Rejected |
| 0.05\_f\_train\_noisy.xlsx | t = -15.687187105022751  p = 1.8444894703799368e-53 | Accepted |
| 0.15\_c\_train\_noisy.xlsx | t = 0.5401741883427151  p = 1.1782181142929824 | Rejected |
| 0.15\_f\_train\_noisy.xlsx | t = -33.882082649189776  p = 2.0499465706172502e-219 | Accepted |
| 0.35\_c\_train\_noisy.xlsx | t = 1.1204862550279004  p = 0.5251570257694668 | Rejected |
| 0.35\_f\_train\_noisy.xlsx | t = -65.00714917229152  p = 0.0 | Accepted |

Ridge

|  |  |  |
| --- | --- | --- |
| **Datasets** | **t-test** | **Null hypothesis** |
| 0.05\_c\_train\_noisy.xlsx | t = 0.7690034423365307  p = 0.8838796000946483 | Rejected |
| 0.05\_f\_train\_noisy.xlsx | t = -14.246736361435117  p = 1.3743354313105911e-44 | Accepted |
| 0.15\_c\_train\_noisy.xlsx | t = 0.6677128331552574  p = 1.008716186802118 | Rejected |
| 0.15\_f\_train\_noisy.xlsx | t = -27.114190712117352  p = 1.1373189211901478e-147 | Accepted |
| 0.35\_c\_train\_noisy.xlsx | t = 1.143105431949681  p = 0.5061358921648065 | Rejected |
| 0.35\_f\_train\_noisy.xlsx | t = -48.12551377781471  p = 0.0 | Accepted |

[Kati paei lathos me ta feature noisy data]

## 6.4. Type of errors

### 6.4.1. Type I

### 6.4.2. Type II

# 7. Conclusions

[to be added]

## 7.1. Repository

Code Is available on GitHub on that link: https://github.com/JoHNNyB92/applied/

[na valw times apo ta idia treximata???]

# References

1. A study of the effect of different types of noise on the precision of supervised learning techniques - David F. Nettleton · Albert Orriols-Puig · Albert Fornells, <https://link.springer.com/article/10.1007/s10462-010-9156-z>
2. Noisy and Missing Data Regression: Distribution-Oblivious Support Recovery - Yudong Chen, Constantine Caramanis, http://proceedings.mlr.press/v28/chen13d.pdf
3. The Truth About Linear Regression 36-350, Data Mining 21 October 2009,  <https://www.stat.cmu.edu/~cshalizi/350/lectures/17/lecture-17.pdf>
4. Scattered Data Approximation of Noisy Data via Iterated Moving Least Squares - Gregory E. Fasshauer and Jack G. Zhang, <https://pdfs.semanticscholar.org/9b7d/891601e006b85a3f49ef432f35524aa2a328.pdf>
5. Machine Learning Algorithms, a study of noise sensitivity – Elias Kalapanidas, Nikolaos Avouris, Marian Craciun, Daniel Niagu, <http://delab.csd.auth.gr/bci1/Balkan/356kalapanidas.pdf>
6. A robust hybrid of lasso and ridge regression Art B. Owen Stanford University October 2006, http://statweb.stanford.edu/~owen/reports/hhu.pdf

#### Datasets

1. <https://github.com/rupakc/UCI-Data-Analysis/tree/master/Boston%20Housing%20Dataset/Boston%20Housing>
2. <http://archive.ics.uci.edu/ml/datasets/air+quality>
3. https://github.com/JoHNNyB92/applied/tree/master/nba\_players

#### Useful Links

1. <https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/>
2. <https://codingstartups.com/practical-machine-learning-ridge-regression-vs-lasso/>
3. <https://sci2s.ugr.es/noisydata>
4. <https://www.kellogg.northwestern.edu/faculty/dranove/htm/dranove/coursepages/Mgmt%20469/noisy-variables.pdf>
5. [**https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/**](https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/)
6. <https://courses.washington.edu/b515/l5.pdf>
7. <https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/10/lecture-10.pdf>
8. <https://stattrek.com/regression/slope-test.aspx>
9. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4285373/>
10. http://www.econ.nyu.edu/user/ramseyj/textbook/chapter11.pdf
11. <http://sociology.soc.uoc.gr/genderstats/t-test.pdf>
12. <https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/t-test/#PairedTTest>

<https://towardsdatascience.com/inferential-statistics-series-t-test-using-numpy-2718f8f9bf2f>

<https://towardsdatascience.com/inferential-statistics-series-t-test-using-numpy-2718f8f9bf2f>

<ftp://ftp.cea.fr/pub/unati/people/educhesnay/pylearn_doc/StatisticsMachineLearningPythonDraft.pdf>

<https://plot.ly/python/t-test/>

<http://sociology.soc.uoc.gr/genderstats/t-test.pdf>

<https://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLSResults.t_test.html>

<https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/hypothesis-testing/>

https://stattrek.com/hypothesis-test/paired-means.aspx?Tutorial=AP

<http://connor-johnson.com/2014/02/18/linear-regression-with-python/>

<https://onlinecourses.science.psu.edu/stat501/node/297/>

<http://dept.stat.lsa.umich.edu/~kshedden/Courses/Stat406/Notes/hypothesis_tests.pdf>

https://towardsdatascience.com/inferential-statistics-series-t-test-using-numpy-2718f8f9bf2f