LIÈGE université
Sciences Appliquées

Université de Liège
Faculté des Sciences Appliquées

Project 2: Bias and variance analysis

ELEN0062-1 : Introduction to machine learning

Andrei Galan (s191903)
Tom Moës(s191408)
Jérôme Pierre (s190979)

November 21, 2022

1.1

First, we're asked to prove that linear regression and the KNN method both fit in the following class of estimator:

$$\hat{f}_{LS}(\mathbf{x}) = \sum_{i=1}^{N} w_i(\mathbf{x}; LS_x)\, y^i \tag{1}$$

Let's start introducing the notation that will be used to discuss linear regression.

The matrix of the N p-dimensional inputs $\mathbf{X}$ is built in the following way:

$$\mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & ... & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & ... & x_{2,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & ... & x_{N,p} \end{pmatrix} \tag{2}$$

The vector of the N corresponding outputs and p parameters naturally follows:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \qquad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} \tag{3}$$

In order to train the model, we suppose that $\hat{\mathbf{y}} = \mathbf{X}\beta$. We wish to estimate β in such a way to minimize the residual sum of squares which is then written as follows :

$$RSS = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{N}(y_i - \mathbf{X}_i\beta)^2 \longrightarrow \mathbf{RSS} = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) \tag{4}$$

Minimizing this RSS yields:

$$\frac{\partial RSS}{\partial \beta} = -2\mathbf{X}^T\,(\mathbf{y} - \mathbf{X}\hat{\beta}) = 0 \Longleftrightarrow \hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\,\mathbf{y} \tag{5}$$

The estimators of the y values obtained for the points $\mathbf{X}^*$ in the test set are then:

$$\hat{\mathbf{y}}(\mathbf{X}^*) = \mathbf{X}^*\hat{\beta} = \mathbf{X}^*(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{H}\mathbf{y} \tag{6}$$

$$\longrightarrow \hat{f}_{LS}(\mathbf{x}^*) = \hat{\mathbf{y}}_j(\mathbf{X}^*) = \sum_{i=1}^{N}\mathbf{H}_{j,i}\,\mathbf{y}_i \tag{7}$$

This expression is indeed of the same type as in expression 1

In classification with KNN, the estimator of the point $\mathbf{x}^*$ is the vector of the probabilities belonging to each class C. In the case of n possible classes:

$$\hat{f}_{LS}(\mathbf{x}) = \begin{pmatrix} P(\mathbf{x} \in C_1) \\ P(\mathbf{x} \in C_2) \\ \vdots \\ P(\mathbf{x} \in C_n) \end{pmatrix} \tag{8}$$

The output $\mathbf{y}$ is simply a vector of 0 whose j^{th} component is 1 if the corresponding input is associated to the j^{th} class.

The derivation then directly comes from the algorithm. Indeed, it only consists in detecting the k nearest neighbours and computing the probability by dividing the number of times a specific class appears among those neighbours by k (then the class attributed to the point is the most likely). It can be directly seen that what we just described is exactly reached by defining w_i in the following expression 1.

$$\begin{aligned} \hat{f}_{LS}(\mathbf{x}) &= \sum_{i=1}^{N} w_i(\mathbf{x}; LS_x)\, y^i \\ w_i &= \begin{cases} 1/k & \text{if i} \in \text{k closest neighbours} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{9}$$

In regression, the same logic holds and equation 9 will still be the same. The only difference is that y^i is the value at the point x^i and that $\hat{f}_{LS}(\mathbf{x})$ is then simply the mean of the values of the k closest neighbours.

1.2

In the case of regression trees, the weights are calculated using the outputs of the learning sample, which is already not the case in the expression of the estimator (it depends only on $\mathbf{x}$ and LS_x). In particular, the weights of a regression tree are calculated by minimizing the square-error in the output for each split. It is also called the impurity I, and is determined by the following equations :

$$\begin{aligned} I(LS) &= var_{y|LS}(y) \\ \Delta I(LS, A) &= var_{y|LS}(y) - \sum_a \frac{|LS_a|}{|LS|} var_{y|LS_a}(y) \end{aligned}$$

In those formulas, A is the subset of LS obtained after the split. In order to reduce ΔI, we chose the subsets leading to the minimum of the variance. Since it depends on the output of LS and LS_a, the estimator $\hat{f}_{LS}(\mathbf{x})$ proposed is not suitable for regression trees.

1.3

It is now asked to decompose the expression of the conditional mean square error into a residual error (RE), a bias and a variance term at a fixed point $\mathbf{x_0}$.

$$E_{LS_y|LS_x}\left(E_{y|\mathbf{x}_0}(y-\hat{f}_{LS}(\mathbf{x}_0))^2\right) = RE(\mathbf{x}_0) + bias^2(\mathbf{x}_0) + variance(\mathbf{x}_0) \tag{10}$$

It is also given that each pair $(\mathbf{x}^i, y^i)$ from our data is such that:

$$y^i = h(\mathbf{x}^i) + \epsilon^i \; ; \epsilon^i \sim N(0, \sigma^2) \tag{11}$$

$$\hat{f}_{LS}(\mathbf{x}_0) = \sum_{i=1}^{N} w_i(\mathbf{x_0}, LS_x)\, y^i \tag{12}$$

$$i \in \{1, 2, ..., N\} \text{ and } (\mathbf{x}^i, y^i) \text{ are iid}$$

In order to demonstrate the decomposition (10), we'll need to start with the first expectation[1]:

$$\begin{aligned}
E_{y|\mathbf{x}_0}(y-\hat{f}_{LS}(\mathbf{x}_0))^2 &= E_{y|\mathbf{x}_0}[(y-E_{y|\mathbf{x}_0}(y)) + (E_{y|\mathbf{x}_0}(y)-\hat{f}_{LS}(\mathbf{x}_0))]^2 \\
&= E_{y|\mathbf{x}_0}[(y-E_{y|\mathbf{x}_0}(y))^2 + (E_{y|\mathbf{x}_0}(y)-\hat{f}_{LS}(\mathbf{x}_0))^2] \\
&= var_{y|\mathbf{x}_0}(y) + E_{y|\mathbf{x}_0}(E_{y|\mathbf{x}_0}(y)-\hat{f}_{LS}(\mathbf{x}_0))^2 \\
&= var_{y|\mathbf{x}_0}(y) + E_{y|\mathbf{x}_0}(h(\mathbf{x}_0)-\hat{f}_{LS}(\mathbf{x}_0))^2 \\
&= var_{y|\mathbf{x}_0}(y) + (h(\mathbf{x}_0)-\hat{f}_{LS}(\mathbf{x}_0))^2
\end{aligned}$$

The second term is the only one affected by the second expectation:

$$\begin{aligned}
E_{LS_y|LS_x}[(h(\mathbf{x}_0)-\hat{f}_{LS}(\mathbf{x}_0))^2] &= E_{LS_y|LS_x}[(\hat{f}_{LS}(\mathbf{x}_0) - E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)] + E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)] - h(\mathbf{x}_0))^2] \\
&= E_{LS_y|LS_x}[(\hat{f}_{LS}(\mathbf{x}_0) - E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)])^2] \\
&\quad + E_{LS_y|LS_x}[E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)] - h(\mathbf{x}_0))^2] \\
&= var_{LS_y|LS_x}(\hat{f}_{LS}(\mathbf{x}_0)) + (E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)] - h(\mathbf{x}_0))^2
\end{aligned}$$

Then:

$$E_{LS_y|LS_x}\left(E_{y|\mathbf{x}_0}(y-\hat{f}_{LS}(\mathbf{x}_0))^2\right) = var_{y|\mathbf{x}_0}(y) + var_{LS_y|LS_x}(\hat{f}_{LS}(\mathbf{x}_0)) + (E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)] - h(\mathbf{x}_0))^2$$

The wanted expression is thus obtained. Precisely, we got that:

- $variance = var_{LS_y|LS_x}(\hat{f}_{LS}(\mathbf{x}_0))$
- $bias^2 = (E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)] - h(\mathbf{x}_0))^2$
- $RE = var_{y|\mathbf{x}_0}(y)$

[1]The fully detailed version is available in the annexes

The residual error quantifies how much the output varies from its expectation (Bayes model), the bias measures the distance between the Bayes model and the expectation of the model and the variance term quantifies how much the model output varies from one learning sample to another.

These different expressions can be further simplified by exploiting the information from (11) and (12). Let's also note that the fact that each pair $(\mathbf{x}^i, y^i)$ are i.i.d. means that the covariance between each y^i is null. The following results are then obtained:

$$\begin{aligned}
var_{LS_y|LS_x}(\hat{f}_{LS}(\mathbf{x}_0)) &= var_{LS_y|LS_x}\left(\sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x)y^i\right) && \text{(13a)}\\
&= \sum_{i=1}^{N} var_{LS_y|LS_x}\left(w_i(\mathbf{x}_0; LS_x)y^i\right) && \text{(13b)}\\
&= \sum_{i=1}^{N} w_i^2(\mathbf{x}_0; LS_x) var_{LS_y|LS_x}(y^i) && \text{(13c)}\\
&= \sum_{i=1}^{N} w_i^2(\mathbf{x}_0; LS_x)\sigma^2 && \text{(13d)}\\
&= \sigma^2 \sum_{i=1}^{N} w_i^2(\mathbf{x}_0; LS_x) && \text{(13e)}
\end{aligned}$$

$$\begin{aligned}
var_{y|\mathbf{x}_0}(y) &= var_{y|\mathbf{x}_0}(h(\mathbf{x}_0) + \epsilon) && \text{(14a)}\\
&= var_{y|\mathbf{x}_0}(h(\mathbf{x}_0)) + var_{y|\mathbf{x}_0}(\epsilon) = \sigma^2 && \text{(14b)}
\end{aligned}$$

$$\begin{aligned}
E_{LS_y|LS_x}(\hat{f}_{LS}(\mathbf{x}_0)) &= E_{LS_y|LS_x}\left(\sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x)y^i\right) && \text{(15a)}\\
&= \sum_{i=1}^{N} E_{LS_y|LS_x}\left(w_i(\mathbf{x}_0; LS_x)y^i\right) && \text{(15b)}\\
&= \sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x) E_{LS_y|LS_x}\left(y^i\right) && \text{(15c)}\\
&= \sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x) E_{y|\mathbf{x}_0}(h(\mathbf{x}^i) + \epsilon) && \text{(15d)}\\
&= \sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x) h(\mathbf{x}^i) && \text{(15e)}
\end{aligned}$$

$$\begin{aligned}
E_{y|\mathbf{x}_0}(y) &= E_{y|\mathbf{x}_0}\left(h(\mathbf{x}_0) + \epsilon\right) && \text{(16a)}\\
&= E_{y|\mathbf{x}_0}\left(h(\mathbf{x}_0)\right) + E_{y|\mathbf{x}_0}\left(\epsilon\right) = h(\mathbf{x}_0) && \text{(16b)}
\end{aligned}$$

Replacing in the initial equation yields:

- $RE = \sigma^2$
- $variance = \sigma^2 \sum_{i=1}^{N} w_i^2(\mathbf{x}_0; LS_x)$
- $bias^2 = \left(h(\mathbf{x}_0) - \sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x) h(\mathbf{x}^i) \right)^2$

In other words, we have that:

$$E_{LS_y|LS_x}\left(E_{y|\mathbf{x}_0}(y - \hat{f}_{LS}(\mathbf{x}_0))^2\right) = \sigma^2 \left(1 + \sum_{i=1}^{N} w_i^2(\mathbf{x}_0; LS_x)\right) + \left(h(\mathbf{x}_0) - \sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x) h(\mathbf{x}^i)\right)^2$$

1.4

It is now asked to apply this formula in the case of KNN. As explained above, we have that:

$$\hat{f}_{LS}(\mathbf{x}) = \sum_{i=1}^{N} w_i(\mathbf{x}; LS_x)\, y^i$$

$$w_i = \begin{cases} 1/k & \text{if i} \in \text{k closest neighbours} \\ 0 & \text{otherwise} \end{cases}$$

Denoting the set of the k closest neighbours by K, we get that:

$$\sum_{i=1}^{N} w_i^2(\mathbf{x}_0; LS_x) = \sum_{i \in K} w_i^2(\mathbf{x}_0; LS_x) = \sum_{i=1}^{k} \frac{1}{k^2} = \frac{1}{k^2} \sum_{i=1}^{k} 1 = \frac{1}{k} \tag{17}$$

$$\sum_{i=1}^{N} w_i(\mathbf{x}_0; LS_x) h(\mathbf{x}^i) = \sum_{i \in K} w_i(\mathbf{x}_0; LS_x) h(\mathbf{x}^i) = \frac{1}{k} \sum_{i=1}^{k} h(\mathbf{x}^i) \tag{18}$$

And then the conditional mean square error turns into:

$$E_{LS_y|LS_x}(E_{y|\mathbf{x}_0}(y - \hat{f}_{LS}(\mathbf{x}_0))^2) = \frac{k+1}{k}\sigma^2 + \left(h(\mathbf{x}_0) - \frac{1}{k} \sum_{i=1}^{k} h(\mathbf{x}^i)\right)^2 \tag{19}$$

We can see that

1. The residual error is still σ^2 which is independent from k. As explained in the last question, this was predictable because it characterizes the Bayes model.
2. The variance takes the form $\frac{\sigma^2}{k}$. This means that it decreases when k gets bigger. It can be understood intuitively because when k increases, it means that the size of the group of points that we considered to find an estimated output a specific x increases. It means that local variations will have less and less impact on the result when k is higher which leads to a lower variance.

3. The squared bias becomes $\left(h(\mathbf{x}_0) - \frac{1}{k}\sum_{i=1}^{k} h(\mathbf{x}^i)\right)^2$. We can start by observing this is simply the square of the subtraction of the Bayes model at $\mathbf{x}_0$ and the mean of the k closest neighbours of $\mathbf{x}_0$. As k increases, the bias term also increases, since the expected value from the k neighbours draws away from the Bayes model.

It can be observed that the bias and the variance have opposite behaviours when k is modified. This is an example of the famous bias-variance tradeoff.

1.5

It is now asked to compare the expression of the the conditional mean square error averaged over LS_x and the one corresponding to the regular decomposition of the expected error.

We then have to compare the two following expressions:

$$\begin{gathered} E_{LS_x}[E_{LS_y|LS_x}\left(E_{y|\mathbf{x}_0}(y - \hat{f}_{LS}(\mathbf{x}_0))^2\right)] \\ E_{LS_x,LS_y}\left(E_{y|\mathbf{x}_0}(y - \hat{f}_{LS}(\mathbf{x}_0))^2\right) \end{gathered} \tag{20}$$

However since $E_A[E_{B|A}\{...\}] = E_{A,B}\{...\}$, both expressions should be equal.

$$\begin{aligned} E_{LS_x}[E_{LS_y|LS_x}\left(E_{y|\mathbf{x}_0}(y - \hat{f}_{LS}(\mathbf{x}_0))^2\right)] &= E_{LS_x,LS_y}\left(E_{y|\mathbf{x}_0}(y - \hat{f}_{LS}(\mathbf{x}_0))^2\right) \\ \iff E_{LS_x}[var_{y|\mathbf{x}_0}(y) + var_{LS_y|LS_x}(\hat{f}_{LS}(\mathbf{x}_0)) + (E_{LS_y|LS_x}[\hat{f}_{LS}(\mathbf{x}_0)] - h(\mathbf{x}_0))^2] \\ = var_{y|\mathbf{x}_0}(y) + var_{LS_x,LS_y}(\hat{f}_{LS}(\mathbf{x}_0)) + (E_{LS_x,LS_y}[\hat{f}_{LS}(\mathbf{x}_0)] - h(\mathbf{x}_0))^2 \end{aligned}$$

We can first see that in both cases, the first term [2] are the same because $var_{y|\mathbf{x}_0}(y)$ is independent from the learning set LS_x. It is only related to the Bayes model.

$$E_{LS_x}[var_{y|\mathbf{x}_0}(y)] = var_{y|\mathbf{x}_0}(y)$$

Concerning the variance term, it can be understood intuitively that the second expression will be larger than the first one. Indeed, in the first expression, $(var_{LS_y|LS_x}(\hat{f}_{LS}(\mathbf{x}_0)))$, the inputs of the learning set are fixed and only the corresponding outputs can vary. This is repeated over different LS_x in order to do the average. In the second expression, $(var_{LS_y,LS_x}(\hat{f}_{LS}(\mathbf{x}_0)))$, the expectation is over the full learning set. Both inputs and corresponding outputs vary. This yields a larger number of possible scenarios and then a higher variance than in the first case. In the first case, since we try to represent a specific situation, the different outputs corresponding to a given input won't change too much and then the conditional variance induces less variability in the learning sets.

[2] which corresponds to the noise

It can also be proven mathematically by using the law of total variance. Let X and Y be two random variables on the same probability space, then the law of total variance says that:

$$var(Y) = E[var(Y|X)] + var(E[Y|X])$$

Replacing Y by $\hat{f}$ and X by LS_x, we directly get that:

$$var_{LS}(\hat{f}) = E_{LS_x}[var_{LS_y|LS_x}(\hat{f})] + \Delta \qquad \Delta \geq 0$$

Regarding the bias, we can intuitively see that it will become smaller. Indeed, since the joint expectation is linked with a larger variability in the learning sets, it will allow the algorithm to have a better estimation of the mean value. The bias term is then going to be smaller in the expression of the joint expectation.

It can also be seen mathematically. Indeed, since both expressions in (20) are the same and that the residual error term are also the same, the increase in the variance term automatically induce a smaller bias term for the second expression.

To sum up, the residual term stays the same, the variance term is larger, and the bias term is smaller in the expression of the joint expectation.

Empirical analysis

We are now asked to perform an analysis of the California housing dataset.

2.1

Let's start by observing that estimating the residual error will be quite hard in this setting.

The residual error has the form $var_{y|\mathbf{x}}(y)$ which is hard to estimate because it is a variance for fixed input points. Indeed, given the finite size of our dataset of continuous values, it is extremely unlikely to have an important number of output for the same input. The fact that our learning set has a high dimension makes it even more unlikely. In the extreme majority of cases, we'll have a single output for each input which makes it really difficult to compute $var_{y|\mathbf{x}}(y)$.

2.2

It is possible to estimate the expected error, the variance and the sum of the residual error and the square bias though.

In order to do so, it is required to get different learning sets for a same test set. This will allow us to see the difference between the prediction corresponding to the same input. Our protocol consists in the following steps:

1. Choose the size of the test set and the one of the learning set and then select them in the whole dataset in such a way that they don't overlap. We don't want to check the prediction from data used to train the model. In the case of this project, we choose a learning set of 15000 instances. Most of the time, we'll use a test set of 5000 instances (see later).
2. Decompose the learning set in smaller ones whose size has to be determined.
3. Using those small training sets, train the model and memorize the prediction it yields for the inputs from the test set.
4. From those predictions, it is possible to compute the wanted values.

   Firstly, it is possible to estimate the expected error by using the mean square error (MSE) of the different predicted values $\hat{Y}$ corresponding to the same input. If we got N of them:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_j - \hat{Y}_i)^2 \tag{21}$$

   Doing so (for each predictions corresponding to a specific input) yields a vector of estimates of the MSE of the size of the test set.

   Since a one-number estimation is wanted, we can simply average over the estimates of the MSE.

   The same methodology can be applied to estimate the variance. The first step consists in evaluating the variance of the predictions for each input from the test set and then average the different variances.

5. Those two values allow to compute the sum of the square bias and the residual error easily by using the decomposition explained in the theoretical part (see 10). We indeed have that:

$$bias^2 + RE = E_{LS_y|LS_x}\left(E_{y|\mathbf{x}}(y - \hat{f}_{LS}(\mathbf{x}_0))^2\right) - variance$$

2.3

In order to express the effect of the complexity of the model on those three values, it is necessary to repeat the previous protocol for each value of the parameter describing the complexity. In our case, we will fix the size of the small learning sets to 500 instances.

Generally speaking, we should observe an increase in variance and a drop of bias as the complexity of the model increases. Indeed, as the model gets more and more complex, it starts to better fit our learning data which leads to more possible outputs for each inputs of the test set and thus a higher variance. At some point, it might even start fitting the noise in our data which will significantly increase the error of the model. This phenomenon is called overfitting. As the complexity increases, the mean output of the model also gets closer to the real one which leads to a lower bias. On the other hand, as a model gets simpler, the bias gets bigger and bigger (and the variance gets smaller). At some point, the model can even become too simple. The corresponding increase in bias will then significantly increase the error and prevent the model from being accurate. This is called underfitting.

Both underfitting and overfitting are nocive to the generalization error of the model. The fact that we cannot decrease as much as wanted the bias and the variance simultaneously is called the bias-variance tradeoff.

Since the expected error is the sum of the bias and the variance, it is expected to first decrease with the decrease of bias and then increase because of the variance.

KNN

Let's start with KNN. The latter is a method which takes into account the different values of nearby points in order to make a prediction. More precisely, it will average the values of the k nearest neighbours in order to make a prediction.

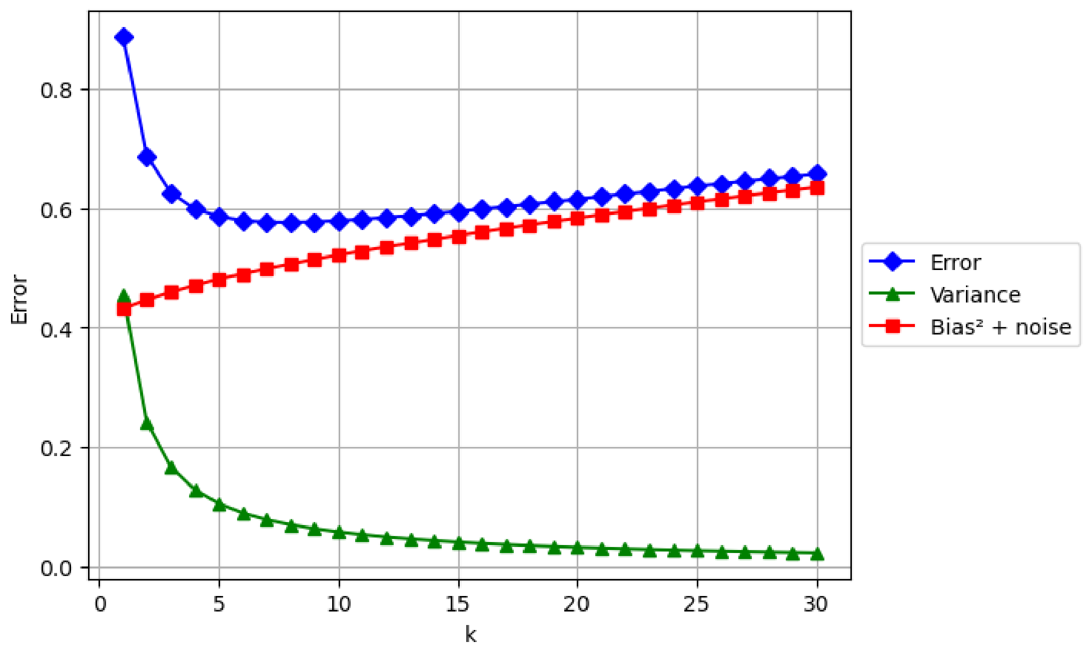

Figure 1: Bias, variance and MSE curves for KNN

It can be seen in the figure above that as k increases, the bias tends to increase and the variance tends to decrease. Since k is a measure of the inverse of the complexity, it is actually exactly what was predicted in the first paragraphs of this section.

Indeed, k represents the number of neighbours taken into account in order to make predictions. If k is really small, the model will get extremely complex in order to correctly predict every value given in the learning set. This is a clear case of overfitting. On the other hand, if k is large, we are averaging the values of the different points in an area around the point at which we want to predict. If k is too large, the model might get too simple and induce underfitting. In the extreme case where k is the total number of inputs in the learning set, this model yields the same value for any input from the test set. It might also be interesting to point out that the variance in such an extreme case is not zero because it depends on the learning set (it will reach some constant).

Those trends were already predicted in section 1.4.

Regression tree

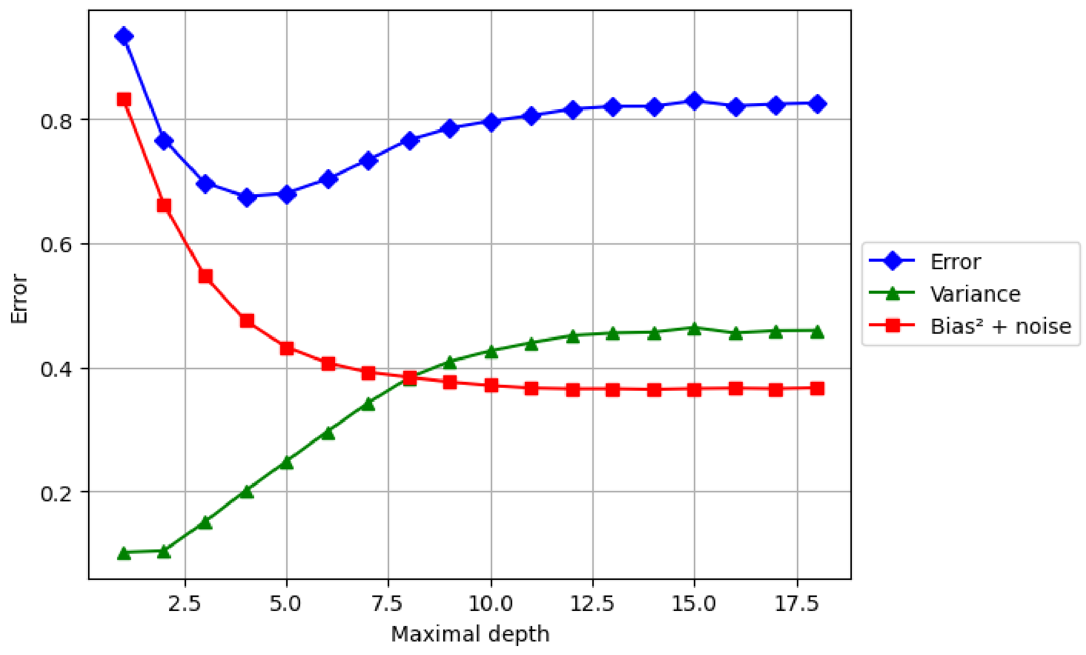

Figure 2: Bias, variance and MSE curves for regression trees

Once again, everything follows what was said in the introduction. Indeed, the maximal depth of the tree is a measure of the complexity of the model. This can be understood easily. With a maximum depth of only 1, the tree makes a single split in the learning set and fails to include details that the learning set may have such as a more complex curve that can't be approximated by a single line (low complexity). On the other hand, even though using the maximal depth possible will allow to perfectly predict the learning set up to the smallest detail (high complexity), it won't greatly predict the outputs of the test set because the noise of the learning sample is fit by the model.

Ridge regression

Ridge regression is a modified version of the classic regression algorithm. Instead of simply minimizing the residual sum of squares (RSS), it wants to add the product of lambda by the sum of the squared coefficient to this loss function. It will allow ridge regression to penalize large coefficients β.

$$\sum_{i=1}^{N}(y_i - \hat{f}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \qquad ; \lambda \in [0; \infty[\tag{22}$$

Let's note that that minimizing this loss function is equivalent to the following constrained minimization problem:

$$argmin(RSS) \qquad \text{s.t} \sum_{\text{j}=1}^{\text{p}} \beta_{\text{j}}^2 < 1/\lambda \tag{23}$$

We can see that the larger λ gets, the smaller the sum of the square of the coefficient must be in order to be selected to minimize the loss function. They (the β's) have then less flexibility in the values they can take. This induces less variability and a higher bias. On the other hand, if λ gets really small, the β's have more possible values to possibly take which directly leads to higher variance and lower bias.

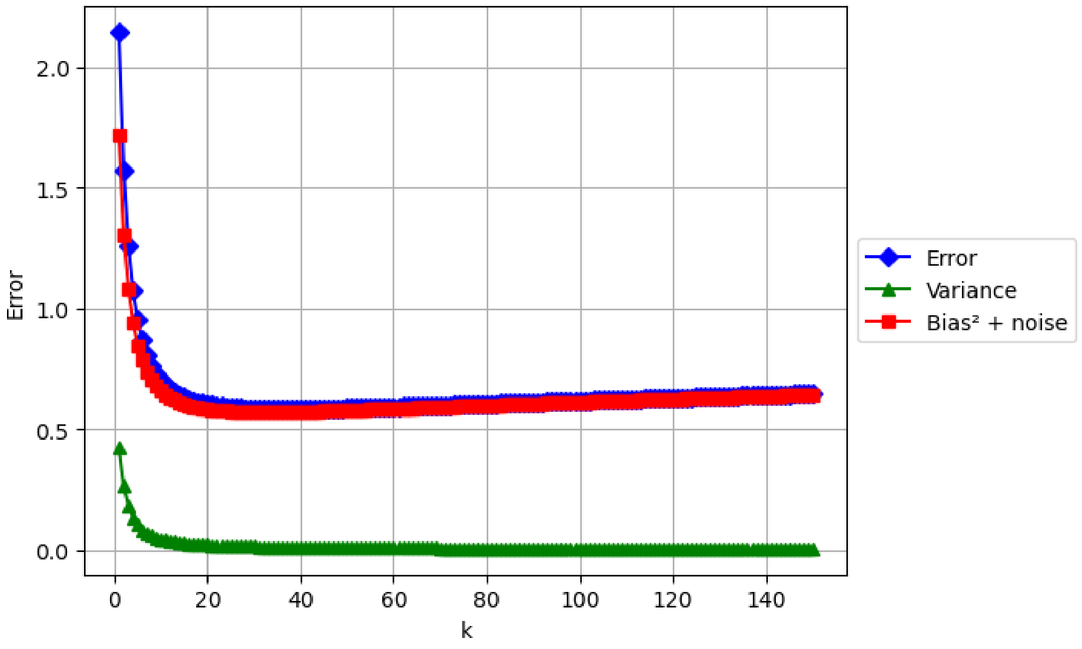

Figure 3: Bias, variance and MSE curves for ridge regression

Even though, we observe that the trends asymptotically correspond to the theoretical ones, they are absolutely not respected for low values of lambda. Such high values of bias for low λ are not expected. Sometimes, theoretical results aren't respected in practice.

We can also observe that these three curves can significantly change according to the chosen test set. Indeed, until now we have chosen to work with a test set of 5000 instances. Since we have a pretty big dataset, we considered it would give a great prediction to the expectation over the learning set without penalizing too much the number of prediction (we still have around 15000 instances for learning sets). In the case of ridge regression, taking a test set of 1000 instances and using 19000 instances for producing learning sets would have given the following curve (see Figure 4. The new curves directly show the asymptotic behaviour we obtained before.

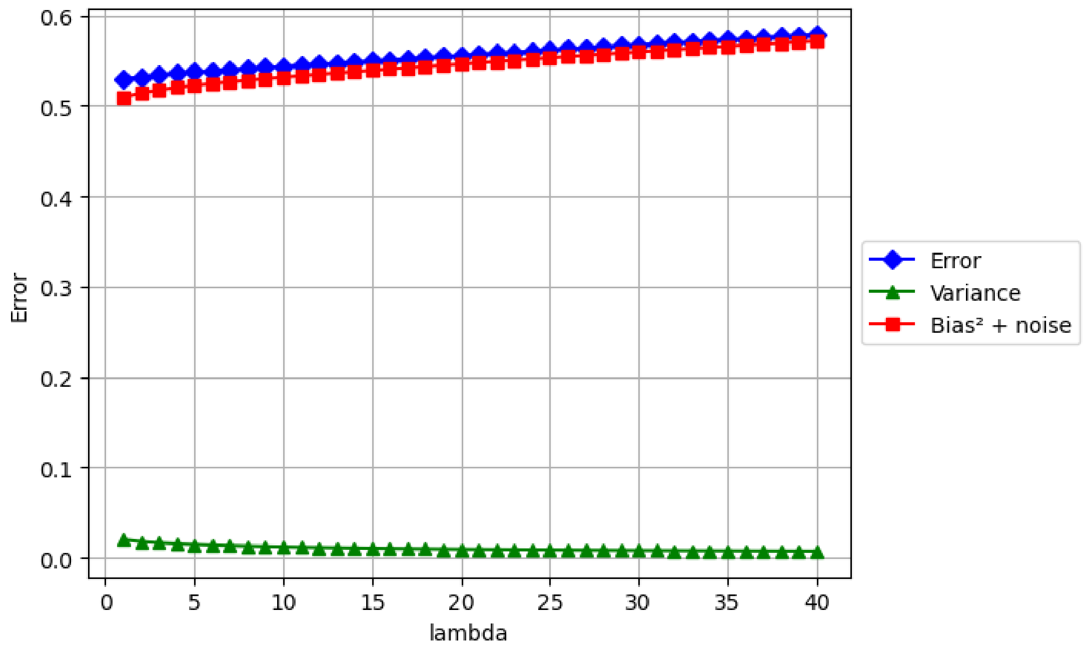

Figure 4: Bias, variance and MSE curves for ridge regression

2.4

It is now interesting to observe the effect of the learning set size on the three different values. In order to do so, we'll simply recompute the different values for an increasing size of the learning sets.

It is important to note that since we have a finite number of instances in the whole dataset, we can't increase the size of the learning set as much as we want. Indeed, at some point, we won't have enough small learning sets to train the model which will results in only a few predictions for each input of the test set. If this number of predictions is too small, we will have poor estimate of the variance. We can't decrease it as much as we want either because at some point, it will make the model estimate the situation pretty poorly (this is explained by the concept of learning curve).

In this section, we used the first 15000 instances to form learning sets. Regarding KNN and regression trees, we used a test set of size 5000. In the case of ridge regression, only the first 3000 instances were used to form the test set.

Regression tree

The complexity of this algorithm is dependent on the size of the learning set which means we'll observe an evolution of all the values. Indeed, it can be seen in figure 9 that the bigger the learning set, the lower the values of bias, variance and expected error. In other words, the model get better with a bigger learning set.

Regarding the variance, such a monotonically decrease is expected when the complexity of the model is independent of the size of the learning samples. When we are dealing with model whose complexity depends on the size of the learning sample, we can expect a loss of bias due the increase in complexity. In this project, the variance of such models will also decrease because the increase in complexity is not big enough to overcome the decrease due to the larger size of the learning samples.

Let's see the effect of the maximal depth on the different curves that can be obtained by modifying the learning set size.

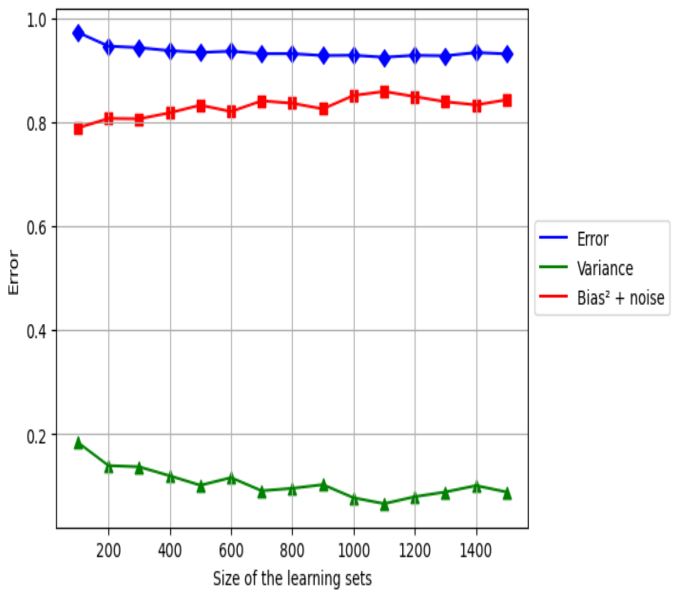

Figure 5: Maximal depth of 1

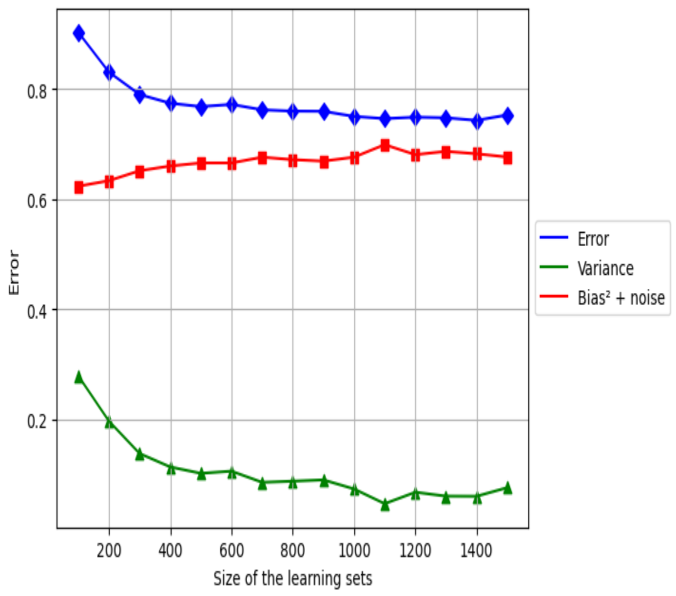

Figure 6: Maximal depth of 2

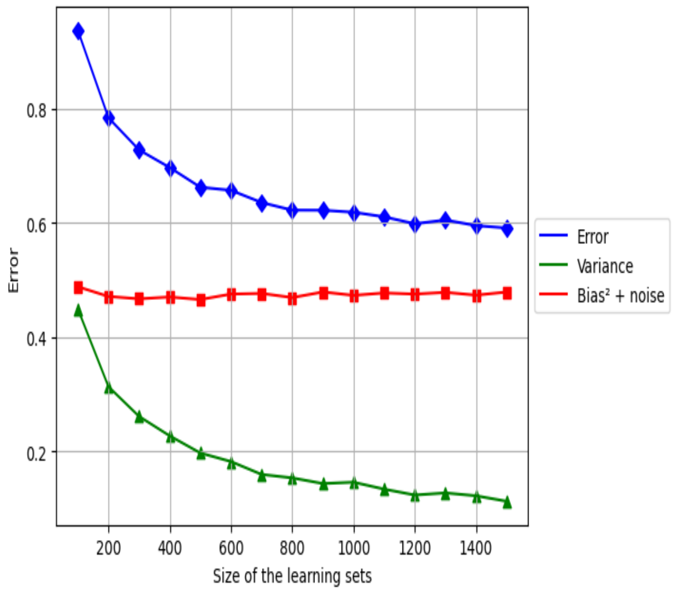

Figure 7: Maximal depth of 4

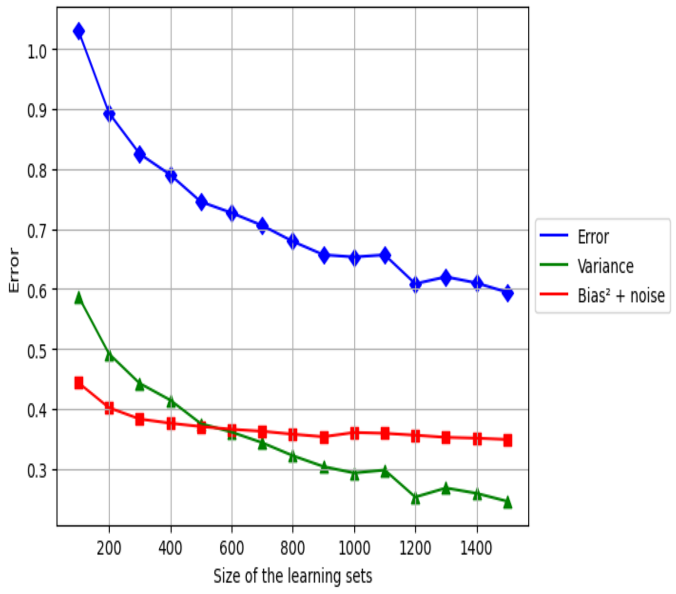

Figure 8: Maximal depth of 8

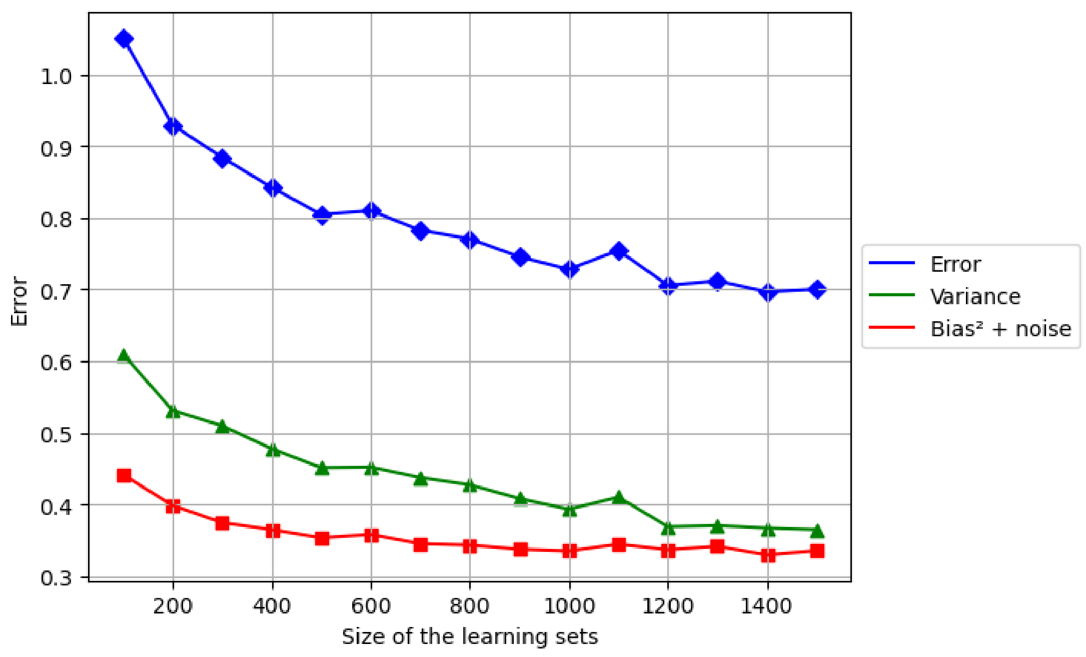

Figure 9: Fully-grown regression tree

It can be observed that these graph express the trend as the curves of Figure 2. Indeed, we can observe that the larger the maximal depth, the smaller the bias and the larger the variance. Concerning the expected error, it is also consistent because a decrease followed by a small increase can be seen. Those comparison are clearly seen by observing the value corresponding to the largest learning sets.

An interesting point is that when the number of splits in the regresion tree is small, we expect to have similar models for different learning sets. This explains why the curves are almost horizontal for a maximal depth of 1[3]. Otherwise, when the model is more complex (the maximal depth is larger), the drops in the different values are bigger.

[3]The small increase in bias can be attributed to randomness in the experimental process. Theoretically, it should decrease, even if it is just a little bit

KNN

The complexity of KNN also depends quite highly on the learning set size. Indeed, let's imagine we are in the case of k=1 and that we want to predict the output at two points that are at the same distance to a given point from the learning set. It means that the predicted output will be the same. However, as soon as the size of the learning set increases, new input points might be inserted between where we wanted to make prediction. The predicted values at those two places might then be different. The complexity has increased.

Once again, the expected error, the bias and the variance decrease with the size of the learning set (see Figure 10 where k is fixed to 5). We got similar graphs as for regression trees

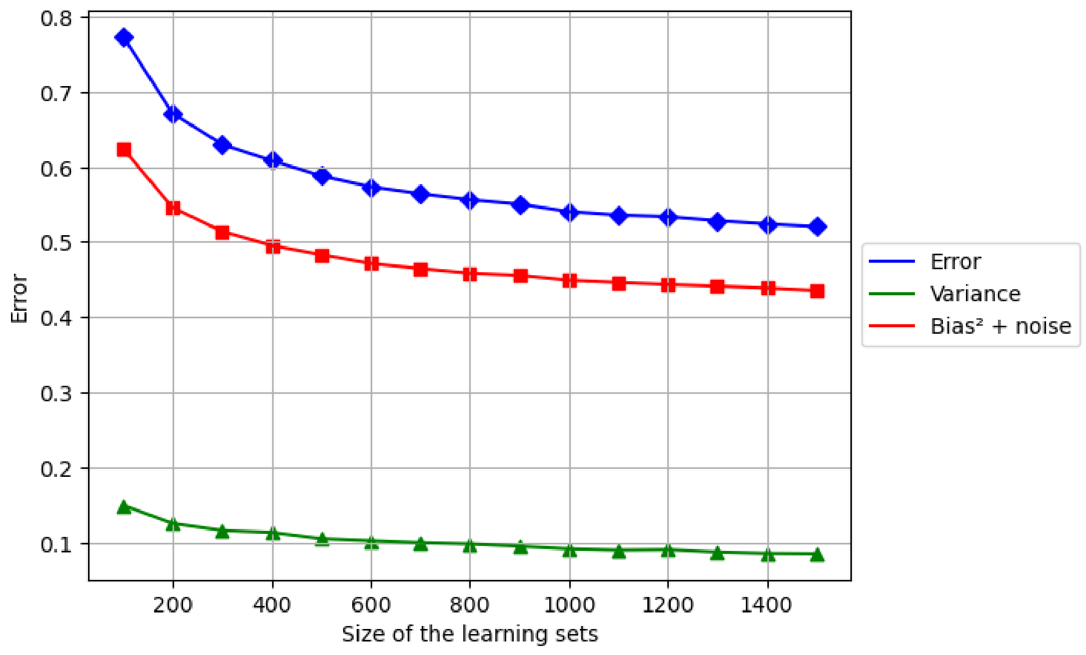

Figure 10: Bias, variance and MSE according to LS size in KNN (k=5)

Ridge regression

The same methodology can be applied to ridge regression as in Figure 11.

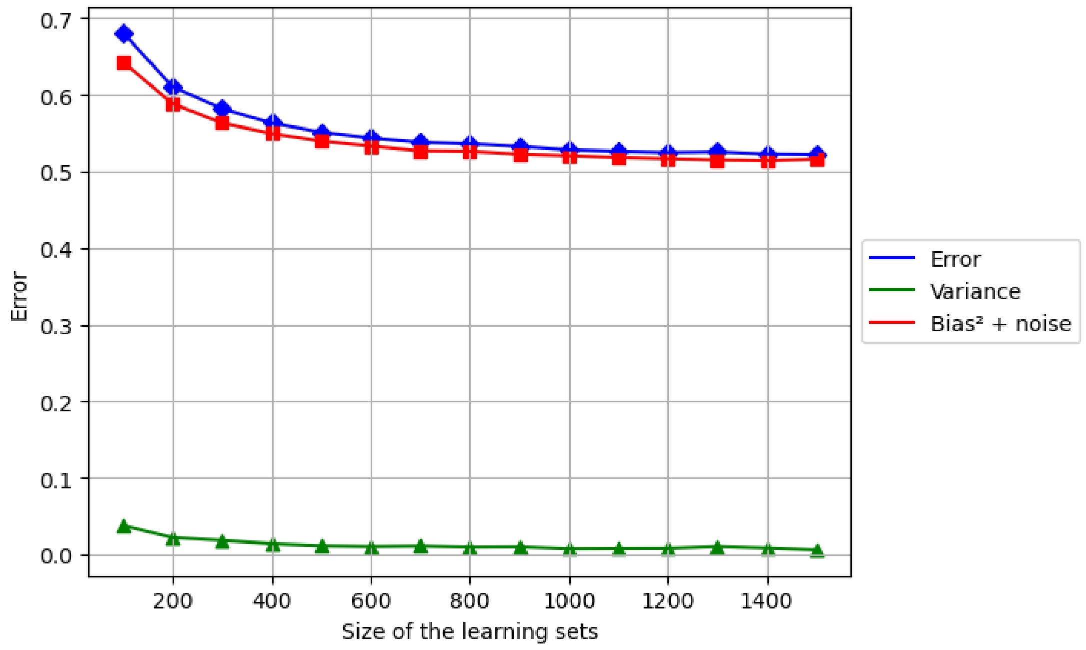

Figure 11: Ridge regression according to LS size with $\lambda = 30$

This graph shows similar results as in the one of KNN or fully grown regression trees: a decrease in every values. This is simply due to the fact that the complexity of ridge regression actually depends on the size of the learning set. Indeed, it can be seen in the expression of the loss function of ridge regression (22), that increasing the number of points will only increase the RSS and not the second term. It then forbids the coefficients from modifying too much under fluctuations in the learning set. The set of possible β (and then of possible hyperplanes) arising from different datasets is way more restricted when the number of points is large.

This can be illustrated quite well by observing two extreme scenarios. Let's start by considering a 2D ridge regression with a learning set of only a few points. It is obvious that small variations in those points can yield very different slopes and interprets. The set of possible lines arising from fluctuations in the learning samples is quite large because the second term of the the loss function (22) can still be dominant. In the other scenario, let's imagine we have an extremely large number of points sampled from a distribution with a high correlation. In this case, any small fluctuation in the learning set won't change the resulting line. If the position of a large majority of points doesn't change under that fluctuation, a small variation in the intercept or the the slope will be significantly penalized by the induced increase in the RSS. The first term of the loss function 22 is greater. The fact that the loss function links the RSS and the coefficient is the reason why the average model won't be the same with respect to the size of the learning set.

Concerning the variance, a decrease is visible. It would be a bit more visible if λ was smaller though. The main reason why the variance doesn't change after a few increases in the size of the learning sets is that it is so small it would be hard to be smaller.

Obviously, λ also plays a role on the curves. As explained before, the larger the λ, the less free the coefficients are. That is why we will observe a great curve for bias when lambda is "large". When λ is null, we are back with the classic linear regression whose complexity is independent from the size of the learning set. This means that the average model will remain the same. This is why we observe a roughly constant bias in the corresponding graph (Figure 12)

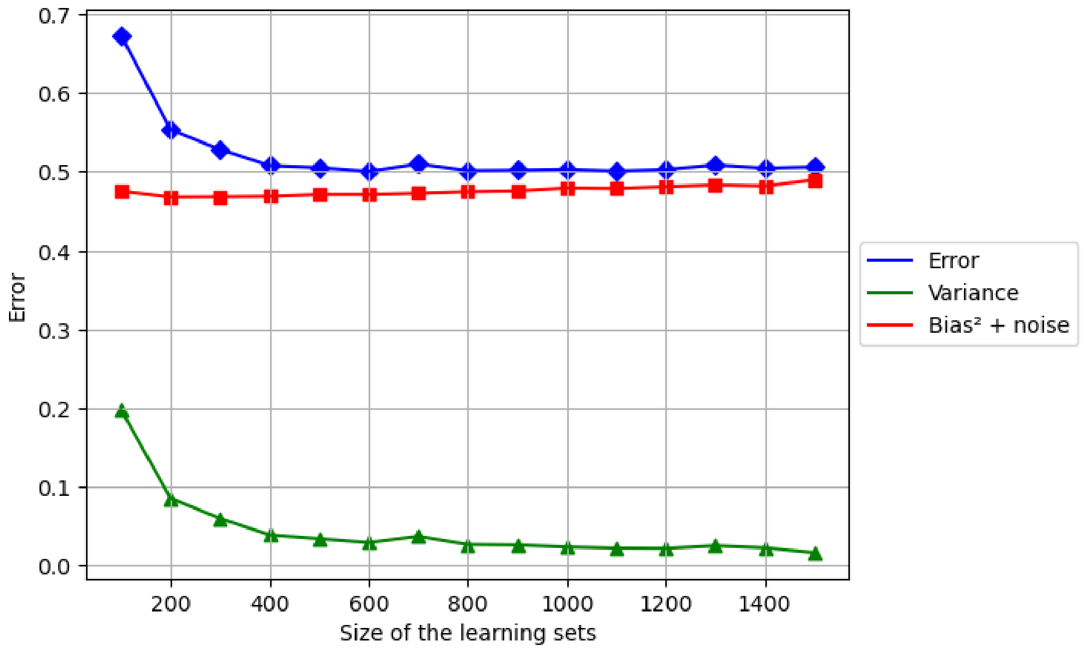

Figure 12: Linear regression

In order to be fully honest, we want to highlight that in this case, the way the test set is constructed may significantly modify the graphs of the ridge regression.

2.5

Bagging (bootstrap aggregating) is a technique which consists in using bootstrap sampling to create multiple learning sets and train the model with them.

We are going to apply this technique to fully grown regression trees and ridge regression ($\lambda = 30$). In order to do so, we'll use the 15000 first instances to build learning sets and the 3000 next for the test set. We'll also use the basic parameter of the scikit-learn implementation. We obtained the following results:

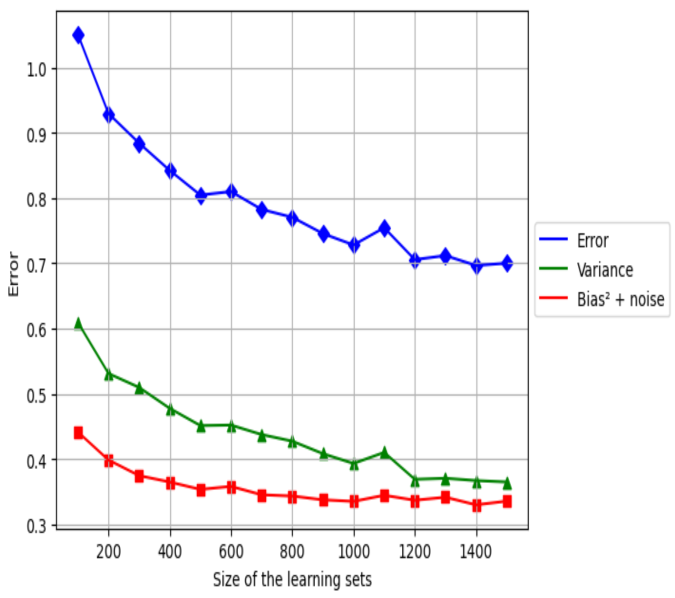

Figure 13: Fully grown regression tree

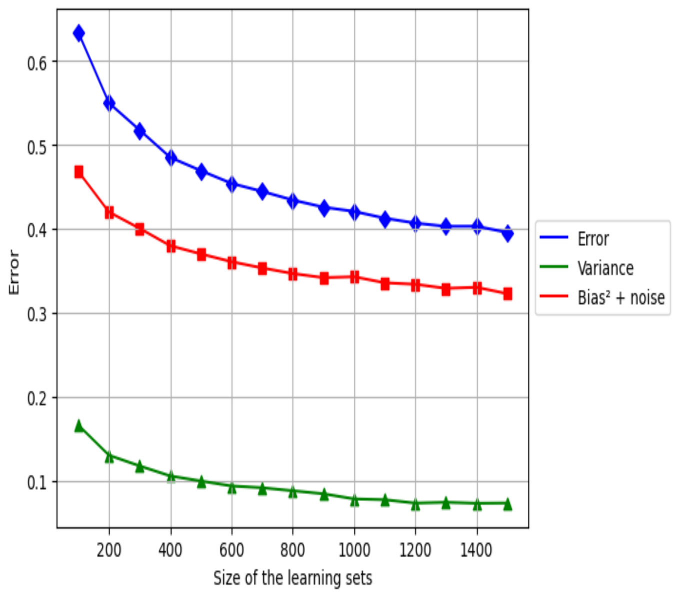

Figure 14: Fully grown regression tree-bagging

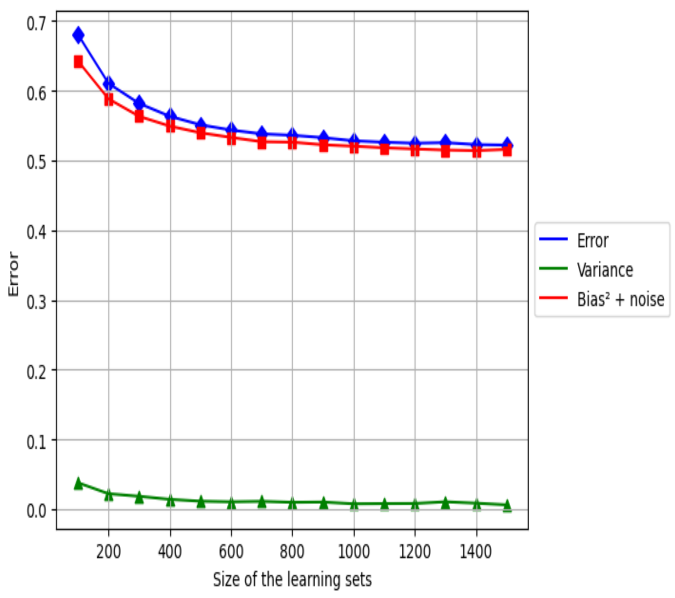

Figure 15: Ridge regression ($\lambda = 30$)

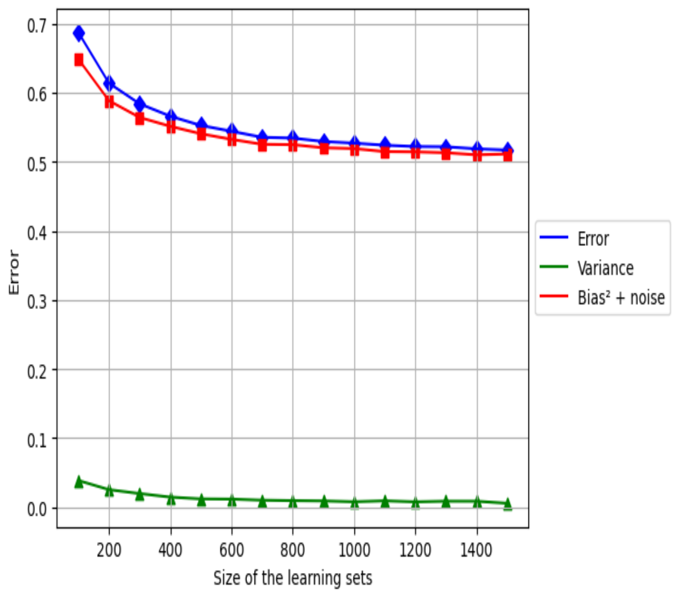

Figure 16: Ridge regression ($\lambda = 30$)-bagging

It can be observed in these figures that bagging significantly decreases the expected error and the variance without altering the bias in the case of regression trees.

In the case of ridge regression, no significant changes are observed.

We can then conclude that bagging is a great technique to decrease variance in the case where it can be "greatly" decreased (high-variance model). However, its impact is negligible when variance is extremely small as in biased model such as ridge regression.

However, since this technique can require quite a lot of time, it might not always be possible to use it on models that often need to go through a learning phase.

2.6

We'll now try to find information about the residual error.

Since it is known that the residual error is a constant, we can compare the different red curve to find an interval containing the wanted value. In order to do so, we just need to find the smallest value of these curves. it will then be the smallest higher bond on the residual error since the red curve represents its sum with the square bias (>0). From Figure 14, we can see that a decent higher bond on the residual error is 0.327.