Information Visualizations Used to Avoid the Problem of Overfitting in Supervised Machine Learning

Robbie T. Nakatsu(✉)

Loyola Marymount University, Los Angeles, CA, USA
rnakatsu@lmu.edu

Abstract. This paper will look at what types of information graphics and visualizations can support supervised Machine Learning tasks: in essence, how to support the problem of model validation and model overfitting. In particular, I look, graphically, at model performance as a function of model complexity. With an appropriate information graphic, we can visualize at what point the model becomes too complex and starts to deteriorate in performance because of model overfitting. I will look at two actual case studies—the first, a regression task using polynomial regression and the second, a classification problem using neural networks. I create information graphics, in particular fitting graphs, to support the end-user in visualizing which model is the best choice.

Keywords: Information visualizations · Machine learning · Data science · Data analytics · Overfitting · Regression tasks · Classification tasks · Data-driven decision making · Regression · Neural networks · Cross validation · Fitting graphs

1 Introduction

Data Science is defined as "a set of fundamental principles that guide the extraction of knowledge from data" [1, p. 2]. In recent years, the discipline has gained prominence as vast amounts of data in almost every industry have become available. Increasingly, businesses and organizations today view data as a strategic asset that can be mined to help them make better decisions. In marketing, for instance, data-mining techniques may be used to better understand target markets, cross-sell to current customers, or otherwise provide better and more-tailored customer service. In general, many organizations today are looking for ways to enable data-driven decision-making in which decisions are based on the analysis of data, rather than intuition alone.

One important area in data science that has garnered attention is **Machine Learning**, a subfield of AI (Artificial Intelligence) that gives computers the ability to learn without being explicitly programmed [2]. Machine Learning involves building predictive models based on the experience of seeing many data examples. In this paper I will look at two types of Machine Learning that involve the prediction of future outcomes: (1) **Regression** problems predict a *continuous-valued* outcome from a large set of data (e.g., predicting future sales of a customer based on historical purchases, and

F.F.-H. Nah and C.-H. Tan (Eds.): HCIBGO 2017, PART II, LNCS 10294, pp. 373–385, 2017.
DOI: 10.1007/978-3-319-58484-3_29

customer characteristics). (2) **Classification** problems predict a *discrete-valued* outcome, or class, from a large historical dataset (e.g., predicting whether an email is spam or not based on seeing an historical dataset of past emails). Both types of tasks are known as **supervised** learning tasks, because they involve training methods in which the targeted outcome is known in advance—hence, the training algorithms are "supervised" or fed the right answer [3].

The difficulty of Machine Learning algorithms is that they are sometimes difficult to understand and use. First, there are a wide variety of techniques such as linear regression, logistic regression, discriminant analysis, decision trees, neural networks, and support vector machines—just to name some of the more popular techniques in use today. Although there exist many "canned" routines in R, and other statistical packages, that are easy to run and use (see e.g., [4]), many users have a difficult time understanding how the algorithms work, let alone whether they are even appropriate for a given task at hand. Many of the algorithms, such as neural networks and support vector machines, are especially problematic because they are black boxes that don't provide an easy-to-understand model on how they work—hence, it is difficult to see how accurate their predictions are.

Second, a central problem in Data Science and Machine Learning is the problem of model overfitting. This refers to "finding chance occurrences in data that look like interesting patterns, but which do not generalize to unseen data" [1, p. 111]. By training a dataset using a Machine Learning algorithm, the data scientist may develop a model that fits well to a current set of data, but does not fit well to future, unseen cases. The solution to this problem is to validate your model on a test dataset, also known as the holdout dataset, because it has been set aside and not used to train the Machine Learning algorithm. I explain the procedure in more detail in Sect. 3.1 of the paper.

In this paper, I illustrate how the model validation task can be supported with information visualizations. The Fitting Graph, notably, is an important information graphic that can be used to visualize model underfit and overfit. I demonstrate the procedure with two tasks: a regression task that predicts the value of a home in the Boston area (Sect. 5), and a classification task that predicts whether a breast tumor is benign or malignant (Sect. 6).

2 A Taxonomy of Supervised Machine Learning Algorithms

Machine Learning algorithms fall into two main categories: (1) supervised and (2) non-supervised. Supervised learning algorithms are the more powerful of the two types because they enable you to predict a target, or outcome variable, from other variables called predictors or features. Hence, I focus this paper on these types of tasks and how well they can predict future cases. Non-supervised learning algorithms can be useful too but not for prediction. One example is clustering (sometimes referred to as segmentation analysis), which attempts to cluster a dataset into homogeneous groups. For example, a marketer may want to identify customers with similar purchasing, or demographic characteristics, so that advertising campaigns can more effectively target these groups.

Within the supervised category, we may distinguish between two types of tasks: (1) regression and (2) classification. While regression attempts to estimate or predict the numeric value of some variable, classification attempts to predict which of a set of classes an individual case belongs to. In both cases predictive accuracy is the benchmark on how well the Machine Learning algorithm is performing. Inference tasks, by contrast, focus on understanding how Y changes as a function of the feature variables (X's). The goal of such tasks is not predictive accuracy, but how well we understand the relationships between the variables—hence, they are sometimes referred to as descriptive rather than predictive. In such cases, model interpretability and simplicity may be a more important criteria for model selection. I do not address inference tasks in this paper.

A central challenge in Machine Learning is choosing the algorithm, or method, for a given application. The choice is not so simple to make. Table 1 shows that there are many different algorithms and techniques at a data scientist's disposal in making the choice. The first point to note is that most of the algorithms work either for regression tasks or classification tasks, but not both—the exceptions in the table are neural networks and support vector machines, which can be used for both task types. Each of the algorithms has relative advantages and disadvantages, but it may be difficult for the data scientist to determine which is the best approach, especially at the outset, when little is known about the nature of the dataset. Furthermore, the field continues to evolve with new and improved learning algorithms, making it difficult to keep track of all the approaches, much less understand which is the best approach.

Table 1. Machine learning algorithms for supervised tasks (regression and classification)

Regression tasks	Classification tasks
Linear regression	Logistic regression
• Ridge regression	Nearest neighbor
• The Lasso	Linear discriminant analysis
Regression splines	Quadratic discriminant analysis
Regression trees	Naïve Bayes
Model trees	Decision trees
Neural networks[a]	Neural networks[a]
Support vector machines[a]	Support vector machines[a]

[a] Denotes Dual Use Algorithms.

Fortunately, for the data scientist, two computing trends have helped with the selection process. The first trend is the dramatic increases in computational power over the last decades, which has made running computationally intensive Machine Learning algorithms more possible. Many more people and organizations have become eager to use Machine Learning techniques, because now they can easily be run on inexpensive computers. Early on, in the 1970s, most of the Machine Learning techniques were linear methods, because fitting non-linear relationships was thought to be computationally infeasible. Today, with the rapid increases in computational power, fitting non-linear models, such as with neural networks, support vector machines,

and polynomial regression, has become much more routine and commonplace [4]. As a result, the data scientist has many more choices in his/her toolkit. Furthermore, cross-validation techniques are no longer computationally prohibitive—in the past, it was infeasible for problems with large n (number of rows in the dataset) and/or large p (number of predictors). The second trend is the development of easy-to-use, canned routines that even the non-technical person, who may lack a sophisticated mathematical or AI background, can master and use. Significantly, R is emerging as the language of choice for the data science community, with new routines and approaches becoming widely available for free. An active community of Machine Learning specialists and R programmers continue to contribute new and improved R routines on Machine Learning and information visualization (see e.g., [5]).

3 The Central Problem of Overfitting

If we develop more and more complex models, we are likely to find patterns in our data. Unfortunately, these patterns may simply be chance occurrences in the data that will not generalize well to unseen cases. The data scientist's end goal, however, should be to develop models that will predict well for cases that have not yet been observed. Finding patterns in your training dataset that do not generalize to unseen cases is broadly known as **overfitting.**

The problem of overfitting is graphically illustrated in Fig. 1 above. I randomly generated a set of x values from a normal distribution with mean 5 and SD = 10. I then calculated y (by applying the quadratic function (y = x2–5x + error), with the error term also randomly generated from a normal distribution with mean 0, SD = 5. Hence, the correct fit for this set of data is quadratic and known in advance. The resulting x–y plot is shown in Fig. 1.

I fitted the data with four different functions using regression:

$$y = \beta_0 + \beta_1 x (\text{linear model}) \tag{1}$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 (\text{quadratic model}) \tag{2}$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \ldots + \beta_{10} x^{10} (\text{polynomial of degree 10}) \tag{3}$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \ldots + \beta_{15} x^{15} (\text{polynomial of degree 15}) \tag{4}$$

As can be seen in Fig. 1, Model (1), the linear model, underfits the data because it does not model the curvature in the data. The linear model is too simple and results in a poor fit (R^2 = 0.269). Model (2) is the correct fit when a quadratic term is added to the regression model, resulting in a smooth curve. The model fit increases dramatically as a result (R^2 = 0.938). Models (3) and (4) overfit the data. Both models chase after the noise in the data, as can be seen in Fig. 1, and result in more erratic curves. When a polynomial of degree 15 is fit to the data, the curve becomes extremely erratic. The R^2 increases slightly to 0.945 and 0.954, respectively for Models (3) and (4), but these

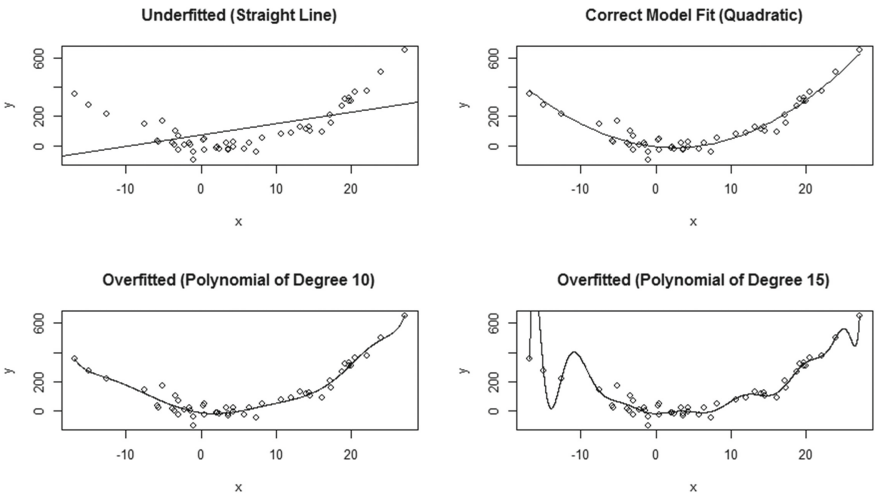

Fig. 1. Four regression models fitted on the same set of data. On the upper left, a simple linear regression underfits the data ($R^2 = 0.269$). On the upper right, the correct quadratic model is fitted ($R^2 = 0.938$). The bottom row shows two overfitted models: on the bottom left, a polynomial of degree 10 is fitted ($R^2 = 0.945$) and on the bottom left a polynomial of degree 15 is fitted ($R^2 = 0.954$).

increases are erroneous and misleading, because they are modeling a better fit in the training data only.

It is easy enough to generate information graphics (in two dimensions) on how a given predictor X influences Y. The non-linearity in the data is apparent in Fig. 1. Furthermore, the overfitting models seem to chase after random noise, also apparent in the bottom two graphs in Fig. 1. However, the determination of model underfitting and overfitting for more complex models involving multiple variables cannot be easily visualized.

3.1 Model Validation: Holdout Evaluation and K-Fold Cross-Validation

This simple example illustrates a central problem in Machine Learning: it is easy to generate more complex models that fit the training data well, but at some point, too much complexity will result in poor performance on data not yet seen. To avoid the problem of overfitting, the model must be validated on a test dataset (or holdout data) that has not been used to train the Machine Learning algorithm.

The procedure for holdout evaluation is simple:

1. Divide your dataset into two samples: a training dataset and a test dataset (an 80/20 split is commonplace in which 80% of the data is the train the model and the remaining 20% is used to validate it).
2. Build separate models on different complexity levels using your training dataset.

3. Validate each model using the test dataset. Test each model, created in step 2, for predictive accuracy on the test dataset. For regression problems, predictive accuracy is defined as the lowest mean error–either mean squared error (MSE) or mean absolute error (MAE). For classification problems, predictive accuracy refers to how well your classifier chooses the right class—or lowest % of misclassified cases (error rate).
4. Choose the model with the highest predictive accuracy—i.e., lowest test error.

There are a few problems with the holdout evaluation method described above. First, even though the test dataset will give us an estimate of predictive accuracy, it is just a single estimate. The single estimate (especially when the sample size is small) may have been a particularly lucky, or unlucky, split between training and test datasets [1]. Second, because we are splitting the data sample into two sets, we are building the model on a smaller set of cases, so we may not generate the best model without use of all of the data.

To address the first issue, a procedure known as **k-fold cross-validation** makes better use of a limited dataset. K-fold cross validation begins by randomly splitting a dataset into k partitions called folds (a typical choice for k is five or ten). K-fold cross validation then iterates k times. In each iteration, a different fold is chosen as the test dataset, while the other k−1 folds are combined to form the training dataset. After iterating k times in this way, an average of the k test errors is calculated so that a more accurate estimate of test error is obtained [1, 3, 4]. Addressing the second issue, once the best model is selected—the one having the lowest average test error—the model can then be re-trained on the entire dataset.

4 Information Visualizations of Overfitting: The Fitting Graph

By visualization, we mean a process by which numerical data and information are converted into meaningful images. A formal definition is given by Spence [6]: "the process of forming a mental model of data, thereby gaining insight into the data."

A fitting graph is an information visualization that illustrates model underfit and overfit. Figure 2 is an example of a typical fitting graph. This figure contains two curves: the top curve represents the test error, and the bottom curve represents the training error. Both curves show how error changes as a function of model complexity. By model complexity, we are referring to the complexity of the predictive model, as specified by the Machine Learning algorithm. The definition of model complexity differs depending on the Machine Learning algorithm used. Here are some examples of how a model can be made more complex in three different Machine Learning contexts:

Regression: adding more X terms, or features, to the model; adding higher-order polynomial terms; adding interaction terms.
Neural Networks: adding more features, or inputs; adding more layers; adding more nodes.
Decision Trees: adding more features, or inputs; having more branches on your decision tree.

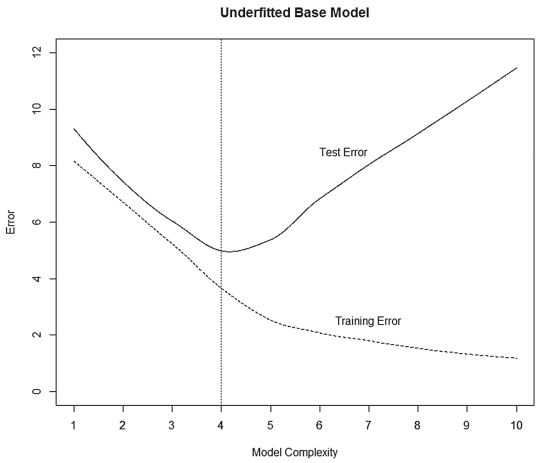

Fig. 2. A fitting graph. The base model (model complexity = 1) is underfitted

Most Machine Learning algorithms are prone to overfitting, so it is important to look at the fitting graph to determine whether the model is overfitted or not. Figure 2 illustrates three key characteristics that apply to all types of fitting graphs. (I use these three characteristics as a check on whether my Machine Learning algorithm is performing as it should, and whether it has been correctly set up):

- The training error curve is always lower than the test error curve for all levels of model complexity.
- The training error curve will decrease as you increase model complexity.
- The test error curve displays a typical "u-shape" in which it decreases first for lower levels of model complexity, and, then, at some point, increases for higher levels of model complexity.

(Note that for the second and third bullet points, the decreases and increases may not occur monotonically due to random fluctuations in your data, as well as way that you are defining "model complexity". In particular, model complexity may not be defined uniformly).

In Fig. 2, the point on the test error curve that represents the minimum test error represents the best model. The vertical line on Fig. 2 indicates where the minimum occurs (model complexity = 4). All models to the left side of the vertical line represent underfitted models, while all models to the right side represent overfitted models.

Figure 3 is an example of another fitting graph. The difference with this graph is that the base model (model complexity = 1) is the correct fit. As evidenced by Fig. 3 the test error starts out at a minimum, and then increases (or remains the same) for higher levels of model complexity. An example is when a simple linear regression, without higher order terms, is the correct fitting model: the data is best fit by a straight line; hence, higher order terms will not decrease the test error.

Finally, Fig. 4 illustrates how a test error curve can plateau, or flatten out, on a portion of the curve. When this happens, the model at the left end of plateau,

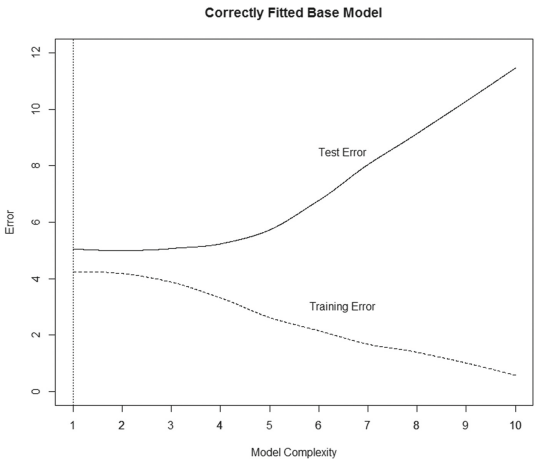

Fig. 3. A fitting graph. The base model (model complexity = 1) is correctly fitted

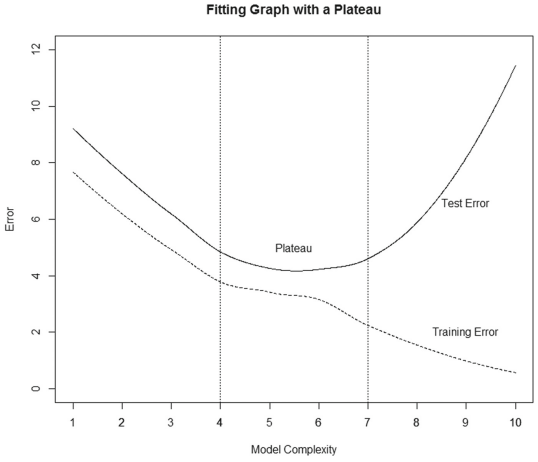

Fig. 4. Fitting graph with a plateau. Between model complexity 4 and 7, the test error doesn't change that much.

representing lower model complexity, may be selected, even if it is not the minimum test error. In Fig. 4, for example, model complexity = 4, may be the selected model, even though complexity levels 5 and 6 result in a lower test error. Simpler, less complex models that are easier to understand may be desired, when more complex models result in only marginal improvements in predictive accuracy. Hence, we may use the fitting graph to assess the tradeoff between test error and model simplicity—plateaus on a test error curve allow us to visualize this.

5 Case I: Regression Task Using Polynomial Regression

For this first example, I used the Boston housing dataset, which is part of the MASS library in R. The dataset contains data on 506 neighborhoods around Boston, Massachusetts. Using polynomial regression, I sought to predict medv (median house values in 000's) from the following five predictors:

1. lstat: percent of households with low socioeconomic status)
2. age: average age of houses
3. rm: (average number of rooms per house)
4. ptratio: pupil/teacher ratio
5. dis: mean distance to five Boston employment centers

The actual datasets contains 13 predictors, but only these five were selected for the analysis. For more information on the Boston dataset, see [7, 8].

To create regression models of varying degrees of complexity, I ran the regression ten times, ranging from a linear fit (polynomial of degree 1) all the way up to a 10^{th} order polynomial fit. For example, the cubic fit (polynomial of degree 3) would include the following predictors in the regression model:

$$\begin{aligned} y = {} & \beta_0 + \beta_{11}\text{lstat} + \beta_{21}\text{age} + \beta_{31}\text{rm} + \beta_{41}\text{ptratio} + \beta_{51}\text{dis} \\ & + \beta_{12}\text{lstat}^2 + \beta_{22}\text{age}^2 + \beta_{32}\text{rm}^2 + \beta_{42}\text{ptratio}^2 + \beta_{52}\text{dis}^2 \\ & + \beta_{13}\text{lstat}^3 + \beta_{23}\text{age}^3 + \beta_{33}\text{rm}^3 + \beta_{43}\text{ptratio}^3 + \beta_{53}\text{dis}^3 \end{aligned}$$

Each of the ten models was trained and validated on a training/set split of 406 training records and 100 test records. To avoid the unreliability inherent in a single estimate of test error, as discussed in Sect. 3.1 above, I randomly generated 10 training/test splits and calculated the average training and test set errors. The **mean absolute error**—or the average absolute difference between the predicted value and the actual value—was calculated and averaged over the 10 randomly generated test samples The results are given in Table 2 and graphically illustrated in Fig. 5.

Table 2. Training error and test error for different levels of model complexity. Model complexity represents degree of polynomial used to fit the regression model.

Model complexity	Training error	Test error
1	3.55	3.63
2	3.00	3.01
3	2.98	3.06
4	2.82	**2.94***
5	2.75	2.94
6	2.69	2.99
7	2.66	3.55
8	2.54	4.18
9	2.48	4.47
10	2.46	9.22

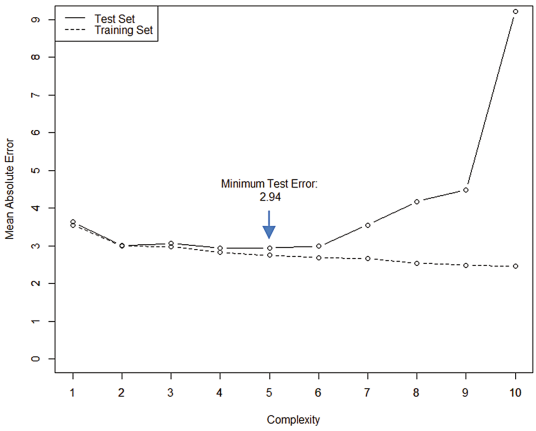

Fig. 5. Fitting graph for regression problem

From the analysis, it appears that a polynomial of degree 4 yields the regression model with the highest predictive accuracy. However, there appears to be a plateau between complexity levels 2 and 6 (the test errors range from 2.94 to 3.01, all equal from a practical standpoint). Hence, the quadratic model (polynomial of degree 2) might be the best choice in terms of balancing model simplicity and predictive accuracy.

6 Case II: Classification Task Using Neural Networks

In the second example, I used the Wisconsin Breast Cancer Diagnostic dataset from the UCI Machine Learning Laboratory [9] to model a classification task, which predicts whether a breast cancer tumor is malignant or benign. The breast cancer dataset includes 569 cases of cancer biopsies with 32 predictors. The diagnosis column has been recoded so that 1 represents a malignant tumor and 0 represents a benign one. To simplify the analysis, I have included only the first ten predictors contained in the dataset, namely, the means of the following biopsy features: radius, texture, perimeter, area, smoothness, compactness, concavity, points, symmetry, and dimension. For more information about this dataset, see [10].

To create neural networks of varying degrees of complexity, I ran the neural network algorithm seven times, varying the number of nodes from one to seven in the hidden layer (the middle layer in Fig. 6). Compare the simple one-node hidden layer model (Fig. 6, left side) to the complex seven-node hidden layer model (Fig. 6, right side).

Each of the seven neural network models was trained and validated on a training/set split of 469 training records and 100 test records. Again, to avoid the unreliability inherent in a single estimate of test error, I randomly generated 10 training/test splits and calculated the average training and test set errors. The errors were calculated as a percent of cases that were misclassified by the neural network. The results are provided in Table 3 and graphically illustrated in Fig. 7.

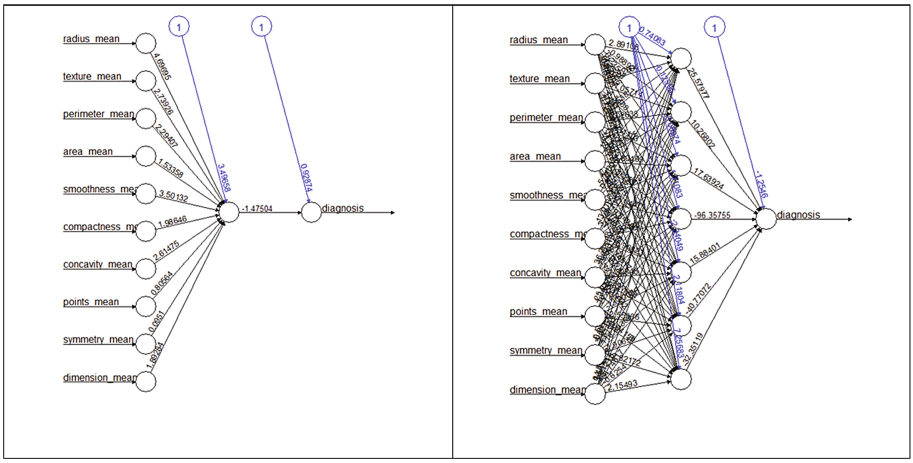

Fig. 6. Two neural networks: the left side has one node in the hidden layer, and the right side has seven nodes in the hidden layer.

Table 3. Training error and test error for different levels of model complexity. Model complexity represents number of nodes in the hidden layer of the neural network.

Model complexity	Training error	Test error
1	8.19%	9.70%
2	1.71%	7.33%
3	1.17%	5.90%
4	1.00%	5.50%
5	0.85%	***4.80%**
6	0.85%	5.70%
7	0.77%	6.00%

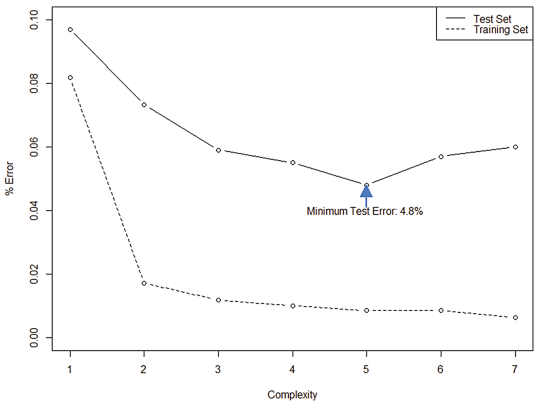

Fig. 7. Fitting graph for classification problem

From the analysis, it appears that the neural network with five nodes in the hidden layer is the best-performing model in terms of accuracy in predicting malignancy of breast tumors. It is interesting to note how the neural network model is able to generate error rates less that 1% on the training data. This points to how prone neural networks are to overfitting. Given enough complexity, the neural network is able to predict the malignancy of breast cancer tumors with close to perfect accuracy on the training dataset. The real test, however, is how well the neural network performs on the test data.

7 Discussion and Conclusions

The future of Machine Learning is promising and exciting. In recent years, we have witnessed an explosion of interest in the field, as more and more methods and techniques are developed by the Machine Learning community. In the era of Big Data, Machine Learning will play a central role in how we gain insights and make sense of complex and massive datasets.

The validation techniques described in the paper are applicable to a wide range of supervised Machine Learning algorithms. What this means is that one can try out a great many types of algorithms, and choose the one that has the highest predictive accuracy on a test dataset. This is especially useful for the end-user who may not have a sophisticated understanding of the mathematics underlying these techniques, but certainly can learn how to apply these cross-validation techniques.

Information visualizations can greatly aid the data scientist in the task of "right-fitting" a model: neither one that is underfit (too simple) nor one that is overfit (too complex). Information graphics like the fitting graph are especially useful in understanding how predictive accuracy varies as a function of model complexity. With an information graphic, we are able to more quickly form a mental model of our data: we are able to visualize how quickly test errors are increasing or decreasing as a function of model complexity. It may happen very quickly, or very slowly—as in Fig. 4, when the test error curve flattens out. The graphic enables us to see this relationship more easily, and assess, for example, trade-offs between model simplicity and model performance.

In this paper, I have simplified the discussion and presented model complexity as a one-dimensional construct. In the regression problem, I represented model complexity by varying the degree of the regression equation: I varied it from 1 (linear regression equation) to 10 (a polynomial regression equation of degree 10). In fact there are other ways of increasing model complexity other than increasing the degree of the polynomial regression equation–namely, the addition of interaction terms, or the addition of more predictors (or Xs). Likewise, the example on neural networks has been simplified. Here, model complexity is represented by the number of nodes in the hidden layer, but I could also have added additional layers to the model, as well as additional input features. The purpose of this simplification was to plot the fitting graph on a single axis of model complexity, and illustrate the relationship between model complexity and predictive accuracy (or error). Without the representation of model complexity as monotonically increasing on the x-axis, it would have been difficult to discern the general patterns in the fitting graph.

References

1. Provost, F., Fawcett, T.: Data Science for Business. O'Reilly Media Inc., Sebastopol (2013)
2. Samuel, A.L.: Some studies in machine learning using the game of checkers. IBM J. Res. Dev. **3**(3), 210–229 (1959)
3. Lantz, B.: Machine Learning in R, 2nd edn. Packt Publishing Ltd., Birmingham (2015)
4. James, G., Witten, D., Hastie, T., Tibshirani, R.: An Introduction to Statistical Learning with Applications in R. Springer, New York (2013)
5. Wickham, H., Grolemund, G.: R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media Inc., Sebastopol (2017)
6. Spence, R.: Information Visualization, vol. 1. Addison-Wesley, New York (2001)
7. Harrison, D., Rubinfeld, D.L.: Hedonic prices and the demand for clean air. J. Environ. Econ. Manag. **5**, 81–102 (1978)
8. Belsey, D.A., Kuh, E., Welsch, R.E.: Regression Diagnostics. Identifying Influential Data and Sources of Collinearity. Wiley, New York (1980)
9. http://archive.ics.uci.edu/ml
10. Mangasarian, O.L., Street, W.N., Wolberg, W.H.: Breast cancer diagnosis and prognosis via linear programming. Oper. Res. **43**, 570–577 (1995)