MACHINE LEARNING-5

1) The residual standard error (RSE) is another statistical term used to describe the difference in standard deviations of observed values versus predicted values as shown by points in a regression analysis. It is a goodness of fit measure that can be used to analyze how well a set of data points fit with the actual model.

RSE is computed by dividing the RSS by the number of observations in the sample less 2, and then taking the square root: RSE = $[RSS/(n-2)]^{1/2}$

2) In statistics, the **explained sum of squares** (**ESS**), alternatively known as the **model sum of squares** or **sum of squares due to regression** (**SSR** – not to be confused with the sum of sqaures (RSS) or sum of squares of errors), is a quantity used in describing how well a model, often a regression model, represents the data being modelled. In particular, the explained sum of squares measures how much variation there is in the modelled values and this is compared to the total sum of squares (TSS), which measures how much variation there is in the observed data, and to the residual sum of sqaures which measures the variation in the error between the observed data and modelled values.

3) Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.
The commonly used regularization techniques are :

L1 regularization

L2 regularization

Dropout regularization

This article focus on L1 and L2 regularization.

A regression model which uses **L1 Regularization** technique is called **LASSO(Least Absolute Shrinkage and Selection Operator)** regression.
A regression model that uses **L2 regularization** technique is called **Ridge regression**.
Lasso Regression adds *"absolute value of magnitude"* of coefficient as penalty term to the loss function(L).

4) Gini Impurity is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree. More precisely, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.

5) Overfitting can be one problem that describes if your model no longer generalizes well.

Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it's possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem. In the case of decision tree's they can learn a training set to a point of high granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behavior of this model that makes it prone to learning every point extremely well — to the point of perfect classification — ie: overfitting.

I recommend the following steps to avoid overfitting:

Use a test set that is not exactly like the training set, or different enough that error rates are going to be easy to see.

What is different enough? Enough to generalize what is being predicted. The problem should dictate this somewhat because if you are predicting on a baseline that is anomalous, you will need to approximate your validation set close enough. If the problem is more general such as spam classification, having enough data will usually have enough entropy to account for enough variance that should exemplify a disparity is cross validation. Additionally, you can use a one-hold-out dataset for extra assurance. You can be more scientific like using "Minimum Description Length principle" which is something related to the size of the error vs the size of the tree but that's getting a bit in the weeds.

Ensure you have enough data.

It's possible that you don't have enough representitive data (more to my first points in this recommendation). There may not be enough examples to describe a specific case. Perhaps you're classifying houses vs apartments and you don't have enough data on apartments within a given range of values such as square feet and bedrooms, the model may not learn that apartments above 2 bedrooms and 2000 square feet in the city can be either house or apartment but is less likely to be an apartement if there are more houses in the dataset than there are in real life, the decision tree will consider the information gain in this range of variables to describe a split on assumptions it can only conclude with the data it observes. I can't think of a better example…

CHECK FOR CLASS IMBALANCE!

- Reduce the complexity of the decision tree model

."

Pruning a tree can be done before or after, but in sklearn, pre-pruning isn't possible, but you can choose to set the minimum amount of data that is required to create a leaf node, which will additionally qualify the conditions on which a split can occur. Additinally, one can set the max-depth to control the complexity, limiting quantity of nodes quite a bit — you could adjust this paramter and check cross-validation each time after you've gridsearched a range that tends to perform well on the classification metric of your choice.

- Use decision trees in an ensemble

Since decision trees are greedy, they are also prone to finding locally optimal solutions without considering a broader assumption about data. This is one reason (in my opinion), that makes these

ideal for ensembles. Ensemble methods (bagging / random forrests, boosting, etc) that allows for weighting different aspects of decision trees (with bootstrapping, with random variable selection, with weighting of weak learners, with sample weight selection.. these are the main aspects that different ensembles mix and match to generalize better)

Reduce the dimensionality of your data

additionally, choose better features. Reducing the complexity of a model can also be done if you reduce the complexity of your data. The potential splits (ie: complexity), can be reduced before you even put your data to the model.

6) Ensemble learning is a technique in machine learning which takes the help of several base models and combines their output to produce an optimized model. This type of machine learning algorithm helps in improving the overall performance of the model. Here the base model which is most commonly used is the Decision tree classifier.

7) Bagging:

Bagging is also known as bootstrap aggregation. It is the ensemble learning method that is generally used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement meaning that the single data points can be selected more than once.

After several data samples are generated, these weak models are trained separately and depend on the element of task regression or classification. For example, the average of those predictions yield a more efficient estimate.

Random Forest is an extension over bagging. It takes one more step to predict a random subset of records. It also creates a random selection of features instead of using all features to develop trees. When it can have several random trees, it is known as the Random Forest.

Bagging has also been leveraged with deep learning models in the finance market, automating critical functions, such as fraud detection, credit risk computations, and option pricing issues.

This research demonstrates how bagging between several machine learning techniques has been leveraged to create loan default risk. This study understands how bagging supports minimizing risk by avoiding credit card fraud within the banking and financial institutions.

Boosting

Boosting is another ensemble process to create a set of predictors. In another terms, it can fit consecutive trees, generally random samples, and at every phase, the objective is to solve net error from the previous trees.

Boosting is generally used to reduce the bias and variance in a supervised learning technique. It defines the family of an algorithm that changes weak learners (base learners) to strong learners. The weak learner is the classifiers that are correct only up to a small extent with the actual classification, while the strong learners are the classifiers that are well correlated with the actual classification.

8) Out-of-Bag Error in Random Forest The out-of-bag error is the average error for each predicted outcome calculated using predictions from the trees that do not contain that data

point in their respective bootstrap sample. This way, the Random Forest model is constantly being validated while being trained.

9) k-Fold Cross-Validation Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into.

10) It is rare that a model will perform at the level you need for production just in the first instance. To find the right solution for your business problem, often you have to go through an iterative cycle. There are multiple pieces that come together to solve the intended machine learning puzzle. You may need to train and evaluate multiple models that include different data setup and algorithms, perform feature engineering a few times or even augment more data. This cycle also involves tweaking your model's **hyperparameters**.

Hyperparameters are the knobs or settings that can be tuned before running a training job to control the behavior of an ML algorithm. They can have a big impact on model training as it relates to training time, infrastructure resource requirements (and as a result cost), model convergence and model accuracy.

Hyperparameters can be broadly divided into 3 categories —

Model hyperparameters — defines the fundamental construct of a model itself for ex. attributes of a neural network architecture like filter size, pooling, stride, padding.

Optimizer hyperparameters — are related to how the model learn the patterns based on data. These types of hyperparameters include optimizers like gradient descent and stochastic gradient descent (SGD), Adam, RMSprop, Adadelta and so on.

Data hyperparameters — are related to the attributes of the data, often used when you don't have enough data or enough variation in data. In such cases data augmentation techniques like cropping, resizing, binarization etc.. are involved.

11) Learning rate is one of The most important thing to consider in whole of machine learning. But choosing the correct learning rate is pretty much impossible all the time.

But…, having the knowledge of what it can do is useful in some cases where we can at least guess what could be the appropriate learning rate.

We will see how even the slightest change in learning_rate can improve the speed drastically or could break your model. I mean it can overshoot the minima and it might never return .

12) *Logistic Regression has traditionally been used as a linear classifier, i.e. when the classes can be separated in the feature space by linear boundaries. That can be remedied however if we happen to have a better idea as to the shape of the decision boundary…*

Logistic regression is known and used as a linear classifier. It is used to come up with a hyper*plane* in feature space to separate observations that belong to a class from all the other observations that do *not* belong to that class. The decision boundary is thus *linear*. Robust and efficient implementations are readily available (e.g. scikit-learn) to use logistic regression as a linear classifier.

13) AdaBoost

AdaBoost or Adaptive Boosting is the first boosting ensemble model. The method automatically adjusts its parameters to the data based on the actual performance in the current iteration. Meaning, both the weights for re-weighting the data and the weights for the final aggregation are re-computed iteratively.

In practice, this boosting technique is used with simple classification tress or stumps as base-learners, which resulted in improved performance compared to the classification by one tree or other single base-learner.

Gradient Boosting

Gradient Boost is a robust ml alogorithm made up of Gradient descent and Boosting. The word 'gradient' implies that you can have two or more derivatives of the same function. Gradient Boosting has three main components: additive model, loss function and a weak learner.

The technique yields a direct interpretation of boosting methods from the perspective of numerical optimisation in a function space and generalises them by allowing optimisation of an arbitrary loss function.

14) Bias Variance Tradeoff

If the algorithm is too simple (hypothesis with linear eq.) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree eq.) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as Trade-off or Bias Variance Trade-off.

This tradeoff in complexity is why there is a tradeoff between bias and variance

15)

The linear, polynomial and RBF or Gaussian kernel are simply different in case of making the hyperplane decision boundary between the classes.

The kernel functions are used to map the original dataset (linear/nonlinear) into a higher dimensional space with view to making it linear dataset.

Usually linear and polynomial kernels are less time consuming and provides less accuracy than the rbf or Gaussian kernels.

The k cross validation is used to divide the training set into k distinct subsets. Then every subset is used for training and others k-1 are used for validation in the entire trainging phase. This is done for the better training of the classification task.

.