Statistics

Multiple Choice Questions

1. Using a goodness of fit,we can assess whether a set of obtained frequencies differ from a set of frequencies.

Ans- d)Expected

2. Chisquare is used to analyse

Ans- d) All of these

3. What is the mean of a Chi Square distribution with 6 degrees of freedom?

Ans- c)6

4. Which of these distributions is used for a goodness of fit testing?
Ans- b) Chisqared distribution

5. Which of the following distributions is Continuous?
Ans- c)F Distribution

6. A statement made about a population for testing purpose is called?
Ans- b)Hypothesis

7. If the assumed hypothesis is tested for rejection considering it to be true is called?
Ans-a) Null Hypothesis

8. If the Critical region is evenly distributed then the test is referred as?
Ans- a)Two tailed

9. Alternative Hypothesis is also called as?
Ans- b)Research Hypothesis

10. In a Binomial Distribution, if ‘n’ is the number of trials and ‘p’ is the probability of success, then the mean value is
given by
Ans- a)np

1. R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of
goodness of fit model in regression and why?

Ans-Both R-squared and Residual Sum of Squares (RSS) are measures of goodness of fit in regression analysis, but they capture different aspects of the model's performance.

R-squared (also known as the coefficient of determination) measures the proportion of variation in the dependent variable that is explained by the independent variables in the model. In other words, it indicates how well the model fits the data, with values ranging from 0 to 1. Higher R-squared values indicate a better fit, as they mean that a larger proportion of the variation in the dependent variable is explained by the independent variables in the model.

On the other hand, RSS measures the total sum of squared differences between the actual values of the dependent variable and the predicted values by the model. It represents the amount of unexplained variation in the data, and lower RSS values indicate a better fit, as they mean that the model is able to explain more of the variation in the data.

Therefore, both measures are useful in evaluating the goodness of fit of a model, but they serve different purposes. R-squared is a useful measure to assess the overall fit of the model and to compare different models, while RSS is useful to identify the degree of the error in the model's predictions.

In general, a good model should have both a high R-squared value and a low RSS value, indicating that it explains a large proportion of the variation in the dependent variable and has a low degree of error in its predictions. However, in some cases, one measure may be more important than the other, depending on the research question and the nature of the data being analyzed.

2. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum
of Squares) in regression. Also mention the equation relating these three metrics with each other.

Ans-In statistical data analysis the **total sum of squares** (**TSS** or **SST**) is a quantity that appears as part of a standard way of presenting results of such analyses. For a set of

observations,yi,i<n , it is defined as the sum over all squared differences between the

observations and their overall mean .:[1] The residual sum of squares (RSS) is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model itself. Instead, it estimates the variance in the residuals, or error term.

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 residual sum of squares (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.

The general regression model with n observations and k explanators, the first of which is a constant unit vector whose coefficient is the regression intercept, is

where y is an $n \times 1$ vector of dependent variable observations, each column of

the $n \times k$ matrix X is a vector of observations on one of the k explanators, is a $k \times 1$ vector of true coefficients, and e is an $n \times 1$ vector of the true underlying errors. The ordinary

least squares estimator for is

The residual vector is , so the residual sum of squares is, after simplification,

Denote as the constant vector all of whose elements are the sample

mean of the dependent variable values in the vector y. Then the total sum of squares is

The explained sum of squares, defined as the sum of squared deviations of the predicted values from the observed mean of y, is

Using in this, and simplifying to obtain , gives the result

that $TSS = ESS + RSS$ if and only if . The left side of this is times

the sum of the elements of y, and the right side is times the sum of the

elements of , so the condition is that the sum of the elements

of y equals the sum of the elements of , or equivalently that the sum of

the prediction errors (residuals) is zero. This can be seen to be true by

noting the well-known OLS property that the $k \times 1$ vector : since the

first column of X is a vector of ones, the first element of this vector is the sum of the residuals and is equal to zero. This proves that the condition holds for the result that $TSS = ESS + RSS$.

In linear algebra terms, we have , , . The proof can be simplified by noting that . The proof is as follows:

Thus,

which again gives the result that *TSS* = *ESS* + *RSS*, since .

3. What is the need of regularization in machine learning?

Ans-Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.

This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "*In regularization technique, we reduce the magnitude of the features by keeping the same number of features.*"

4. What is Gini–impurity index?

Ans-Gini Index, also known as Gini impurity, **calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly.** If all the elements are linked with a single class then it can be called pure.

Let's perceive the criterion of the Gini Index, like the properties of entropy, *the **Gini index varies between values 0 and 1**, where 0 expresses the purity of classification, i.e. All the elements belong to a specified class or only one class exists there. And 1 indicates the random distribution of elements across various classes. The value*

of 0.5 of the Gini Index shows an equal distribution of elements over some classes.

While designing the decision tree, the features possessing the least value of the Gini Index would get preferred. You can learn another tree-based algorithm(Random Forest).

The Gini Index is determined by deducting the sum of squared of probabilities of each class from one, mathematically, Gini Index can be expressed as:

Gini Index Formula

Where Pi denotes the probability of an element being classified for a distinct class.

Classification and Regression Tree (CART) algorithm deploys the method of the Gini Index to originate binary splits.

In addition, ***decision tree algorithms exploit Information Gain to divide a node and Gini Index or Entropy is the passageway to weigh the Information Gain.***

5. Are unregularized decision-trees prone to overfitting? If yes, why?

Ans- Decision trees are prone to overfitting, especially when a tree is particularly deep. This is due to the amount of specificity we look at leading to smaller sample of events that meet the previous assumptions. This small sample could lead to unsound conclusions. An example of this could be predicting if the Boston Celtics will beat the Miami Heat in tonight's basketball game. The first level of the tree could ask if the Celtics are playing home or away. The second level might ask if the Celtics have a higher win percentage than their opponent, in this case the Heat. The third level asks if the Celtic's leading scorer is playing? The fourth level asks if the Celtic's second leading scorer is playing. The fifth level asks if the Celtics are traveling back to the east coast from 3 or more consecutive road games on the west coast. While all of these questions may be relevant, there may only be two previous games where the conditions of tonights game were met. Using only two games as the basis for our classification would not be adequate for an informed decision. One way to combat this issue is by setting a max depth. This will limit our risk of overfitting; but as always, this will be at the expense of error due to bias. Thus if we set a max depth of three, we would only ask if the game is home or away, do the Celtics have a higher winning percentage than their opponent, and is their leading scorer playing. This is a simpler model with less variance sample to sample but ultimately will not be a strong predictive model.

6. What is an ensemble technique in machine learning?

Ans-Ensemble methods are techniques that create multiple models and then combine them to produce improved results. Ensemble methods usually produces more accurate solutions than a single model would. This has been the case in a number of machine learning competitions, where the winning solutions used

ensemble methods. In the popular Netflix Competition, the winner used an ensemble method to implement a powerful collaborative filtering algorithm. Another example is KDD 2009 where the winner also used ensemble methods. You can also find winners who used these methods in Kaggle competitions, for example here is the interview with the winner of CrowdFlower competition.

It is important that we understand a few terminologies before we continue with this article. Throughout the article I used the term "model" to describe the output of the algorithm that trained with data. This model is then used for making predictions. This algorithm can be any machine learning algorithm such as logistic regression, decision tree, etc. These models, when used as inputs of ensemble methods, are called "base models".

In this blog post I will cover ensemble methods for classification and describe some widely known methods of ensemble: voting, stacking, bagging and boosting.

7. What is the difference between Bagging and Boosting techniques?

Ans-The difference between bagging and boosting techniques are:

S.NO	Bagging	Boosting
1.	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2.	Aim to decrease variance, not bias.	Aim to decrease bias, not variance.
3.	Each model receives equal weight.	Models are weighted according to their performance.
4.	Each model is built independently.	New models are influenced by the performance of previously built models.

S.NO	Bagging	Boosting
5.	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6.	Bagging tries to solve the over-fitting problem.	Boosting tries to reduce bias.
7.	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.
8.	In this base classifiers are trained parallelly.	In this base classifiers are trained sequentially.
9	Example: The Random forest model uses Bagging.	Example: The AdaBoost uses Boosting techniques

8. What is out-of-bag error in random forests?

Out-of-bag (**OOB**) **error**, also called **out-of-bag estimate**, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging). Bagging uses subsampling with replacement to create training samples for the model to learn from. OOB error is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.[1]

Bootstrap aggregating allows one to define an out-of-bag estimate of the prediction performance improvement by evaluating predictions on those observations that were not used in the building of the next base learner.

9. What is K-fold cross-validation?

Ans-K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data. K refers to the number of groups the data sample is split into. For example, if you see that the k-value is 5, we can call this a 5-fold cross-validation. Each fold is used as a testing set at one point in the process.

K-fold Cross-Validation Process:

1. Choose your k-value
2. Split the dataset into the number of k folds.
3. Start off with using your k-1 fold as the test dataset and the remaining folds as the training dataset
4. Train the model on the training dataset and validate it on the test dataset
5. Save the validation score
6. Repeat steps 3 – 5, but changing the value of your k test dataset. So we chose k-1 as our test dataset for the first round, we then move onto k-2 as the test dataset for the next round.
7. By the end of it you would have validated the model on every fold that you have.
8. Average the results that were produced in step 5 to summarize the skill of the model.

9. What is hyper parameter tuning in machine learning and why it is done?

Ans-Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors. Note that the learning algorithm optimizes the loss based on the input data and tries to find an optimal solution within the given setting. However, hyperparameters describe this setting exactly.

For instance, if we work on natural language processing (NLP) models, we probably use neural networks, support-vector machines (SVMs), Bayesian networks, and Extreme Gradient Boosting (XGB) for tuning parameters.

10. What issues can occur if we have a large learning rate in Gradient Descent?

Ans-Learning rate is used to scale the magnitude of parameter updates during gradient descent. The choice of the value for learning rate can impact two things: 1) how fast the algorithm learns and 2) whether the cost function is minimized or not. Figure 2 shows the variation in cost function with a number of iterations/epochs for different learning rates.

It can be seen that for an optimal value of the learning rate, the cost function value is minimized in a few iterations (smaller time). This is represented by the blue line in the figure. If the learning rate used is lower than the optimal value, the number of iterations/epochs required to minimize the cost function is high (takes longer time). This is represented by the green line in the figure. If the learning rate is high, the cost function could saturate at a value higher than the minimum value. This is represented by the red line in the figure. If the learning rate selected is very high, the cost function could continue to increase with iterations/epochs. An optimal learning rate is not easy to find for a given problem. Though getting the right learning is always a challenge, there are some well-researched methods documented to figure out optimal learning rates. Some of these techniques are discussed in the following sections. In all these techniques the fundamental idea is to vary the learning rate dynamically instead of using a constant learning rate.

11. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

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.

While logistic regression makes core assumptions about the observations such as IID (each observation is independent of the others and they all have an identical probability distribution), the use of a linear decision boundary is *not* one of them. The linear decision boundary is used for

reasons of simplicity following the Zen mantra – when in doubt simplify. In those cases where we suspect the decision boundary to be nonlinear, it may make sense to formulate logistic regression with a nonlinear model and evaluate how much better we can do. That is what this post is about. Here is the outline. We go through some code snippets here but the full code for reproducing the results can be downloaded from github.

- Briefly review the formulation of the likelihood function and its maximization. To keep the algebra at bay we stick to 2 classes, in a 2-d feature space. A point [x, y] in the feature space can belong to only one of the classes, and the function f (x,y; c) = 0 defines the decision boundary as shown in Figure 1 below.

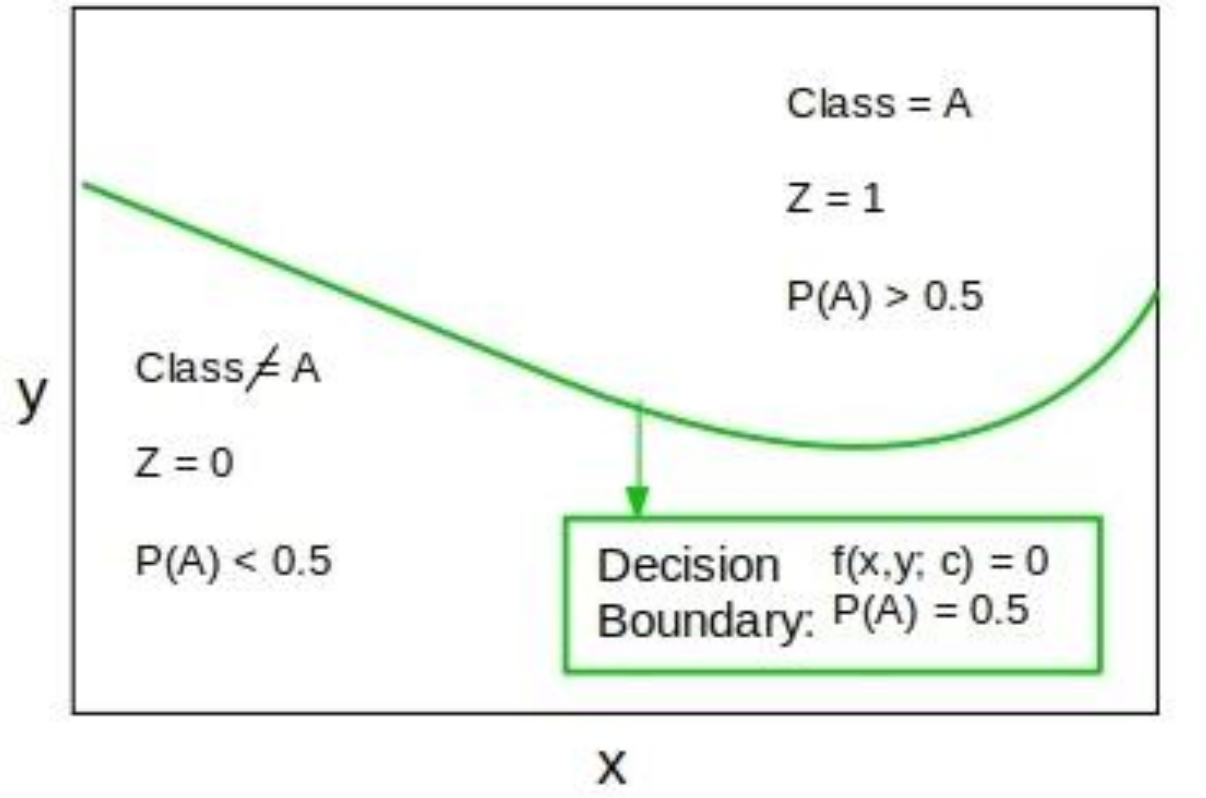

Figure 1. The decision boundary f(x,y; c) = 0 in feature space, separates the observations that belong to class A from those that do not belong to class A. c are the parameters to be estimated.

- Consider a decision boundary that can be expressed as a polynomial in feature variables but linear in the weights/coefficients. This case lends itself to be modeled within the (linear) framework using the API from scikit-learn.
- Consider a generic nonlinear decision boundary that cannot be expressed as a polynomial. Here we are on our own to find the model parameters that maximize the likelihood function. There is excellent API in the scipy.optimize module that helps us out here.

13. Differentiate between Adaboost and Gradient Boosting.

Ans-The difference between Adaboost and Gradient Boosting are:

Loss Function:

The technique of Boosting uses various loss functions. In case of Adaptive Boosting or AdaBoost, it minimises the exponential loss function that can make the algorithm sensitive to the outliers. With Gradient Boosting, any differentiable loss function can be utilised. Gradient Boosting algorithm is more robust to outliers than AdaBoost.

Flexibility

AdaBoost is the first designed boosting algorithm with a particular loss function. On the other hand, Gradient Boosting is a generic algorithm that assists in searching the approximate solutions to the additive modelling problem. This makes Gradient Boosting more flexible than AdaBoost.

Benefits

AdaBoost minimises loss function related to any classification error and is best used with weak learners. The method was mainly designed for binary classification problems and can be utilised to boost the performance of decision trees. Gradient Boosting is used to solve the differentiable loss function problem. The technique can be used for both classification and regression problems.

Shortcomings

In the case of Gradient Boosting, the shortcomings of the existing weak learners can be identified by gradients and with AdaBoost, it can be identified by high-weight data points.

Wrapping Up

Though there are several differences between the two boosting methods, both the algorithms follow the same path and share similar historic roots. Both the algorithms work for boosting the performance of a simple base-learner by

iteratively shifting the focus towards problematic observations that are challenging to predict.

In the case of AdaBoost, the shifting is done by up-weighting observations that were misclassified before, while Gradient Boosting identifies the difficult observations by large residuals computed in the previous iterations.

14. What is bias-variance trade off in machine learning?

Ans-In statistics and machine learning, the **bias–variance tradeoff** is the property of a model that the variance of the parameter estimated across samples can be reduced by increasing the bias in the estimated parameters. The **bias–variance dilemma** or **bias–variance problem** is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:[1][2]

- The *bias* error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- The *variance* is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting).

The **bias–variance decomposition** is a way of analyzing a learning algorithm's expected generalization error with respect to a particular problem as a sum of three terms, the bias, variance, and a quantity called the *irreducible error*, resulting from noise in the problem itself.

15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Ans-SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example ***linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.***
Introduce Kernel functions for sequence data, graphs, text, images, as well as vectors. The most used type of kernel function is **RBF.** Because it has localized and finite response along the entire x-axis.
The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces.