MACHINE LEARNING WORKSHEET SET 5

Q1 to Q15 are subjective answer type questions, Answer them briefly.

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?

A. What is R Squared?

R Squared is used to determine the strength of correlation between the predictors and the target. In simple terms it lets us know how good a regression model is when compared to the average. R Squared is the ratio between the residual sum of squares and the total sum of squares.

$$R^2 = 1 - \frac{SSR}{SST}$$

What Is the Residual Sum of Squares (RSS)?
A 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.

Linear regression is a measurement that helps determine the strength of the relationship between a dependent variable and one or more other factors, known as independent or explanatory variables.

The Formula for the Residual Sum of Squares (RSS)
$RSS = \sum^{n}_{i=1} (y^{i} - f(x_i))^{2}$

where:

- y_i = the ith value of the variable to be predicted
- $f(x_i)$ = predicted value of y_i
- n = upper limit of summation

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.

A. Total sum of squares(TSS)

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 , it is defined as the sum over all squared differences between the observations and their overall mean .

Explained sum of squares (ESS)

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. 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 squares, which measures the variation in the error between the observed data and modelled values.

What Is the Residual Sum of Squares (RSS)?
A 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.

Linear regression is a measurement that helps determine the strength of the relationship between a dependent variable and one or more other factors, known as independent or explanatory variables.

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

A. 1. WHAT IS REGULARIZATION?

Regularization in machine learning terms is to make things acceptable or regular. The process involves the shrinking of data coefficients to tend to zero values. In other words, the process of regularization of the regularization methods in machine learning will discourage overfitting the model, which then learns to be more flexible in a complex environment.

2. HOW DOES REGULARIZATION WORK?

The basic concept of regularization in machine learning is to provide bigger loss values to complex models and award losses to complex models with the introduction of a complexity term.

Take the simple regularization in the logistic regression relationship mentioned below.

Y≈ W_0 W_1 X_1 W_2 X_(2) ⋯ W_P X_P

Here, the value to be predicted or learned relation of regularized logistic regression is Y and its deciding features are given by the values of X_1to P etc., and the weights of the features are represented by W_1to P and the bias by W_0.

To fit this regression model, one also requires the regularization parameter in machine learning based on the weights, bias and loss functions to be able to predict the Y value. Linear regression uses RSS or residual sum of squares as its loss function regularization parameter, which can be denoted by

[RSS= the sigmoid function (∑) of _(j=1)^m (Y_i-W_0-∑_(i=1)^n W_i X_ji)^2] which is also called objective of linear regression without regularization.

The algorithm uses the loss function for learning from the training dataset by adjusting the coefficients or weights. When given noisy datasets, it has overfitting issues, and the estimated coefficients produced do not generalize the unseen data, thus needing regularization. This process of regularization in deep learning causes the larger coefficients to be penalized and pushes the estimates learned to zero value.

3. REGULARIZATION TECHNIQUES

The 2 regularization techniques in machine learning are the Lasso and Ridge Regression techniques for regularization in machine learning, which are different based on the manner of penalizing the coefficients in the L1 and L2 regularization in machine learning.

Lasso Regression (L1 Regularization)

The L1 regularization in the machine learning technique modifies the RSS value. It adds a shrinkage quantity or penalty, which is the sum of the coefficient’s absolute values in the sigmoid equation below where estimated coefficients use the modified loss-function.

∑_(j=1)^m (Y_i-W_0-∑_(i=1)^n W_i X_ji)^2 α∑_(i=1)^n |W_i |=RSS α∑_(i=1)^n |W_i |

Lasso Regression or lasso regularization hence uses for normalization of the absolute values of coefficients and hence differs from ridge regression since its loss function is based on the weights or absolute coefficients. The algorithm for optimization will now inflict a penalty on high coefficients in what is called the L1 norm. The value of alpha-α is similar to the ridge regression regularization tuning parameter and is a tradeoff parameter to balance out the RS coefficient’s magnitude.

If $\alpha=0$, we have a simple linear regression. If $\alpha=\infty$, the coefficient of lasso regression is zero. If values are such that $0<\alpha<\infty$, the coefficient has a value between 1 and 0. It appears very similar to ridge regression, but let’s have a look at both techniques with a different perspective. In ridge regression, the coefficients or sum of squares of weights is equal or less than s, meaning the equations 2 parameters can be expressed as $W_1^2 W_2^2 \leq s$.

Hence, coefficients in ridge regression have the least loss function for all within the circle points of the equation’s solution. In lasso regression, the coefficients are the sum of modulus of weights being equal to or less than s. Then, we get the equation$|W_1| |W_2| \leq s$

The coefficients of ridge regression coefficients have the smallest values of loss function for all diamond points lying within the diamond of the equation’s solution.

Ridge Regression (L2 Regularization)

This regularization in machine learning technique uses L2 regularization or modifies the shrinkage quantity of the RSS through a penalty added in which is given by the square of the coefficient’s magnitude and represented as modified loss function as below

$$\sum_{j=1}^m \left(Y_i-W_0-\sum_{i=1}^n W_i X_{ji}\right)^2 \alpha\sum_{i=1}^n W_i^2=RSS\ \alpha\sum_{i=1}^n W_i^2$$

Here, alpha-α is the parameter for shrinkage quantity called the tuning parameter, and this value decides the penalty on the model. ie the emphasis of the tuning parameter is used.

If $\alpha=0$, the penalty is zero, and we have simple linear regression. If $\alpha=\infty$, the ridge regression coefficient value is zero since the modified loss function’s core loss function is ignored. The square of coefficients becomes minimized, causing the value of the parameter to tend to zero. If $0<\alpha<\infty$, the value of the coefficient lies between 1 and 0. Hence selecting values of the α coefficient is critical. This ridge regression regularization technique is called the L2 norm.

4. What is Gini–impurity index?

A. Gini Impurity

`Gini Impurity` is a measurement of the likelihood of an **incorrect classification** of a new instance of a random variable, if that new instance were **randomly classified according to the distribution of class labels** from the data set.

Gini impurity is **lower bounded by 0**, with 0 occurring if the data set contains only one class.

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

A. Overfitting in decision trees

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

There's a great quote for this: "Everything should be made as simple as possible, but no simpler."

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. What is an ensemble technique in machine learning?

A. INTRODUCTION TO ENSEMBLE METHODS

Ensemble method in Machine Learning is defined as the multimodal system in which different classifier and techniques are strategically combined into a predictive model (grouped as Sequential Model, Parallel Model, Homogeneous and Heterogeneous methods etc.) Ensemble method also helps to reduce the variance in the predicted data, minimize the biasness in the predictive model and to classify and predict the statistics from the complex problems with better accuracy.

Types of Ensemble Methods in Machine Learning

Ensemble Methods help to create multiple models and then combine them to produce improved results, some ensemble methods are categorized into the following groups:

1. Sequential Methods

In this kind of Ensemble method, there are sequentially generated base learners in which data dependency resides. Every other data in the base learner is having some dependency on previous data. So, the previous mislabeled data are tuned based on its weight to get the performance of the overall system improved.

Example: Boosting

2. Parallel Method

In this kind of Ensemble method, the base learner is generated in parallel order in which data dependency is not there. Every data in the base learner is generated independently.

Example: Stacking

3. Homogeneous Ensemble

Such an ensemble method is a combination of the same types of classifiers. But the dataset is different for each classifier. This will make the combined model work more precisely after the aggregation of results from each model. This type of ensemble method works with a large number of datasets. In the homogeneous method, the feature selection method is the same for different training data. It is computationally expensive.

Example: Popular methods like bagging and boosting comes into the homogeneous ensemble.

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

A.

Bagging Vs Boosting

We all use the Decision Tree Technique on day to day life to make the decision. Organizations use these supervised machine learning techniques like Decision trees to make a better decision and to generate more surplus and profit.

Ensemble methods combine different decision trees to deliver better predictive results, afterward utilizing a single decision tree. The primary principle behind the ensemble model is that a group of weak learners come together to form an active learner.

There are two techniques given below that are used to perform ensemble decision tree.

Bagging

Bagging is used when our objective is to reduce the variance of a decision tree. Here the concept is to create a few subsets of data from the training sample, which is chosen randomly with replacement. Now each collection of subset data is used to prepare their decision trees thus, we end up with an ensemble of various models. The average of all the assumptions from numerous tress is used, which is more powerful than a single decision tree.

Random Forest is an expansion over bagging. It takes one additional step to predict a random subset of data. It also makes the random selection of features rather than using all features to develop trees. When we have numerous random trees, it is called the Random Forest.

These are the following steps which are taken to implement a Random forest:

- Let us consider **X** observations **Y** features in the training data set. First, a model from the training data set is taken randomly with substitution.
- The tree is developed to the largest.

- The given steps are repeated, and prediction is given, which is based on the collection of predictions from n number of trees.

Advantages of using Random Forest technique:

- It manages a higher dimension data set very well.
- It manages missing quantities and keeps accuracy for missing data.

Disadvantages of using Random Forest technique:

Since the last prediction depends on the mean predictions from subset trees, it won't give precise value for the regression model.

Boosting:

Boosting is another ensemble procedure to make a collection of predictors. In other words, we fit consecutive trees, usually random samples, and at each step, the objective is to solve net error from the prior trees.

If a given input is misclassified by theory, then its weight is increased so that the upcoming hypothesis is more likely to classify it correctly by consolidating the entire set at last converts weak learners into better performing models.

Gradient Boosting is an expansion of the boosting procedure.

1. Gradient Boosting = Gradient Descent + Boosting

It utilizes a gradient descent algorithm that can optimize any differentiable loss function. An ensemble of trees is constructed individually, and individual trees are summed successively. The next tree tries to restore the loss (It is the difference between actual and predicted values).

Advantages of using Gradient Boosting methods:

- It supports different loss functions.
- It works well with interactions.

Disadvantages of using a Gradient Boosting methods:

- It requires cautious tuning of different hyper-parameters.

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

A. Out-of-bag error:
After building the classifiers (S trees), for each (Xi,yi) in the original training set i.e. T, select all Tk which does not include (Xi,yi). This subset, pay attention, is a set of bootstrap datasets which do not contain a particular record from the original dataset. This set is called out-of-bag examples. There are n such subsets (one for each data record in original dataset T). OOB classifier is the aggregation of votes ONLY over Tk such that it does not contain (xi,yi). The out-of-bag estimate for the generalization error is the error rate of the out-of-bag classifier on the training set (compare it with known yi's).

The study of error estimates for bagged classifiers gives empirical evidence to show that the out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need for a set-aside test set.

9. What is K-fold cross-validation?

A. ***k-fold cross-validation***

is one of the most popular strategies widely used by data scientists. It is a ***data partitioning strategy*** so that you can effectively use your dataset to build ***a more generalized model***. The main intention of doing any kind of machine learning is to develop a more generalized model which can perform well on ***unseen data***. One can build a perfect model on the training data with 100% accuracy or 0 error, but it may fail to generalize for unseen data. So, it is not a good model. It overfits the training data. Machine Learning is all about ***generalization*** meaning that model's performance can only be measured with data points that have never been used during the training process. That is why we often split our data into a training set and a test set.

Data splitting process can be done more effectively with k-fold cross-validation. Here, we discuss two scenarios which involve k-fold cross-validation. Both involve splitting the dataset, but with different approaches.

- Using k-fold cross-validation for evaluating a model's performance
- Using k-fold cross-validation for hyperparameter tuning

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

A. Introduction to Hyperparameter Tuning

Data Science is made of mainly two parts. Data analytics and machine learning modeling. Although Data Science has a much wider scope, the above-mentioned components are core elements for any Data Science project. Let me quickly go through the difference between data analytics and machine learning.

For any Data Science project, having historical data is a key essence. This data might not be in the format that makes sense, or we might need to process the data to get insights from the same. To achieve insights, the below process is important.

1. **Data Analytics** – Historical data has a lot to say, so to hear what it has in store for us, we need to analyze it thoroughly. We can get insights into how variables are related to each other, which variable adds what value to the data, do we have any missing value, do we have any extreme (outlier) values, etc. Data description, data pre-processing, data munching, data cleaning, and exploratory data analysis all come under one umbrella, i.e**., Data Analytics**.
2. **Machine Learning**– Post doing data analytics, these insights should be used in the most sought-after way to predict the future values. How to do that? To answer this, we have machine learning models. When a machine learns on its own based on data patterns from historical data, we get an output which is known as a machine learning model. In a broad category, machine learning models are classified into two categories, **Classification**, and **Regression**.

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as **Hyperparameters.** These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. Of course, you must select from a specific list of hyperparameters for a given model as it varies from model to model.

Often, we are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as **Hyperparameter Tuning.**

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

A. An Introduction to Gradient Descent and Linear Regression

Gradient descent is one of those "greatest hits" algorithms that can offer a new perspective for solving problems. Unfortunately, it's rarely taught in undergraduate computer science programs. In this post I'll give an introduction to the gradient descent algorithm, and walk through an example that demonstrates how gradient descent can be used to solve machine learning problems such as linear regression.

At a theoretical level, gradient descent is an algorithm that minimizes functions. Given a function defined by a set of parameters, gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values that minimize the function. This iterative minimization is achieved using calculus, taking steps in the negative direction of the function gradient.

It's sometimes difficult to see how this mathematical explanation translates into a practical setting, so it's helpful to look at an example. The canonical example when explaining gradient descent is linear regression.

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

A. **Logistic regression** is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, pregnant, etc.) or 0 (FALSE, failure, non-pregnant, etc.).

The goal of logistic regression is to find the best fitting (yet biologically reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a *logit transformation* of the probability of presence of the characteristic of interest:

$$\text{logit}(p)=b0+b1X1+b2X2+b3X3+\ldots+bkXk$$

where p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds:

$$\text{odds} =p1-p=\text{probability of presence of characteristicprobability of absence of characteristic}$$

and

$$\text{logit}(p)=\ln(p1-p)$$

Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.

13. Differentiate between Adaboost and Gradient Boosting.

A. 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 trees 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 machine learning algorithm 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.

The Comparison

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.

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

A. Bias-Variance Trade off – Machine Learning

It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine learning algorithm. There is a tradeoff between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of **Regularization** constant. Proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

Bias

The bias is known as the difference between the prediction of the values by the ML model and the correct value. Being high in biasing gives a large error in training as well as testing data. Its recommended that an algorithm should always be low biased to avoid the problem of underfitting.

By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as **Underfitting of Data**. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.

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

A.Introduction to SVM

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

Working of SVM

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).

Implementing SVM in Python

- For implementing SVM in Python − We will start with the standard libraries import as follows −

SVM Kernels

In practice, SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. The following are some of the types of kernels used by SVM.

Linear Kernel

It can be used as a dot product between any two observations. The formula of linear kernel is as below −

$$K(x,xi)=sum(x*xi)K(x,xi)=sum(x*xi)$$

From the above formula, we can see that the product between two vectors say x & xi is the sum of the multiplication of each pair of input values.

Polynomial Kernel

It is more generalized form of linear kernel and distinguish curved or nonlinear input space. Following is the formula for polynomial kernel −

$$k(X,Xi)=1+sum(X*Xi)^{\wedge}d$$k(X,Xi)=1+sum(X∗Xi)^d

Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

Radial Basis Function (RBF) Kernel

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically −

$$K(x,xi)=exp(-gamma*sum(x-xi^{\wedge}2))$$K(x,xi)=exp(−gamma∗sum(x−xi^2))

Here, *gamma* ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of *gamma* is 0.1.

As we implemented SVM for linearly separable data, we can implement it in Python for the data that is not linearly separable. It can be done by using kernels.