

MACHINE LEARNING

Worksheet Set - 6

In Q1 to Q5, only one option is correct, Choose the correct option:

1. In which of the following you can say that the model is overfitting?

ANS: C) High R-squared value for train-set and Low R-squared value for test-set.

2. Which among the following is a disadvantage of decision trees?

ANS: B) Decision trees are highly prone to overfitting.

3. Which of the following is an ensemble technique?

ANS: C) Random Forest

4. Suppose you are building a classification model for detection of a fatal disease where detection of the disease is most important. In this case which of the following metrics you would focus on?

ANS: A) Accuracy

5. The value of AUC (Area under Curve) value for ROC curve of model A is 0.70 and of model B is 0.85. Which of these two models is doing better job in classification?

Ans: C) both are performing equal

In Q6 to Q9, more than one options are correct, Choose all the correct options:

6. Which of the following are the regularization technique in Linear Regression??

ANS: A) Ridge, D) Lasso

7. Which of the following is not an example of boosting technique?

ANS: A) Adaboost, D) Xgboost.

8. Which of the techniques are used for regularization of Decision Trees?

ANS: D) All of the above

9. Which of the following statements is true regarding the Adaboost technique?

ANS: A) We initialize the probabilities of the distribution as $1/n$, where n is the number of data-points

B) A tree in the ensemble focuses more on the data points on which the previous tree was not performing well

Q10 to Q15 are subjective answer type questions, Answer them briefly.

10. Explain how does the adjusted R-squared penalize the presence of unnecessary predictors in the model?

ANS: The adjusted R-squared compensates for the addition of variables and only increases if the new predictor enhances the model above what would be obtained by probability. Conversely, it will decrease when a predictor improves the model less than what is predicted by chance.

R² shows the linear relationship between the independent variables and the dependent variable. It is defined as $1 - \frac{SSE}{SSTO}$ which is the sum of squared errors divided by the total sum of squares. $SSTO = SSE + SSR$ which are the total error and total sum of the regression squares. As independent variables are added SSR will continue to rise (and since SSTO is fixed) SSE will go down and R² will continually rise irrespective of how valuable the variables you added are.

The Adjusted R² is attempting to account for statistical shrinkage. Models with tons of predictors tend to perform better in sample than when tested out of sample. The adjusted R² "penalizes" you for adding the extra predictor variables that don't improve the existing model. It can be helpful in model selection. Adjusted R² will equal R² for one predictor variable. As you add variables, it will be smaller than R².

$$\text{Adj } R^2 = 1 - \left(\frac{(n-1)}{(n-k-1)} \right) (1 - R^2)$$

Where k = # of independent variables, n = # observations

While adjusted R² says the proportion of the variation in your dependent variable (Y) explained by **more than 1** independent variables (X) for a linear regression model.

11. Differentiate between Ridge and Lasso Regression.

ANS: **Lasso** is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights. Thus, the absolute values of weight will be (in general) reduced, and many will tend to be zeros. During training, the objective function become:

$$\sum_{i=1}^n (y - Xw)^2 + \alpha \sum_{j=1}^p w_j^2$$
$$\frac{1}{2m} \sum_{i=1}^m (y - Xw)^2 + \alpha \sum_{j=1}^p |w_j|$$

As you see, Lasso introduced a new hyperparameter, α , the coefficient to penalize weights.

(2) **Lasso and ElasticNet tend to give sparse weights (most zeros)**, because the l_1 regularization cares equally about driving down big weights to small weights, or driving small weights to zeros. If you have a lot of predictors (features), and you suspect that not all of them are that important, Lasso and ElasticNet may be really good idea to start with.

Ridge takes a step further and penalizes the model for the sum of squared value of the weights. Thus, the weights not only tend to have smaller absolute values, but also really tend to penalize the extremes of the weights, resulting in a group of

weights that are more evenly distributed. The objective function becomes:

ElasticNet is a hybrid of Lasso and Ridge, where both the absolute value penalization and squared penalization are included, being regulated with another coefficient `l1_ratio`:

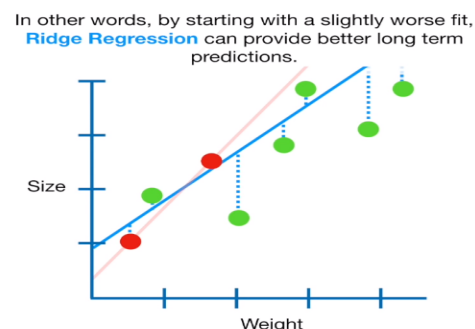
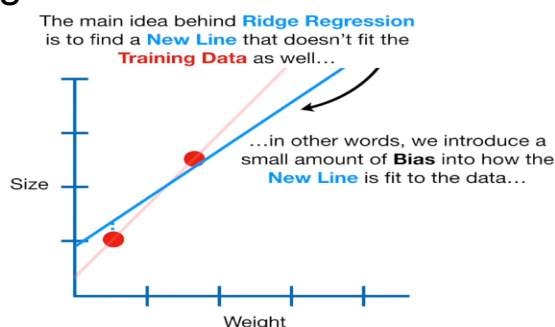
Ridge tends to give small but well distributed weights, because the l_2 regularization cares more about driving big weight to small weights, instead of driving small weights to zeros. If you only have a few predictors, and you are confident that all of them should be really relevant for predictions, try Ridge as a good regularized linear regression method.

L2 Ridge Regression
It is a Regularization Method to reduce Overfitting. We try to use a trend line that overfit the training data, and so, it has much higher variance than the OLS. The main idea of Ridge Regression is to fit a new line that doesn't fit the training data. In other words, we introduce a certain Amount of Bias into the new trend line.

What we do in practice, is to introduce a Bias that we call Lambda, and the Penalty Function is: $\lambda \cdot \text{slope}^2$.

$$\frac{1}{2m} \sum_{i=1}^m (y - Xw)^2 + \alpha \cdot \text{ratio} \cdot \sum_{j=1}^p |w_j| + 0.5 \cdot \alpha \cdot (1 - \text{ratio}) \cdot \sum_{j=1}^p w_j^2$$

The Lambda is a penalty terms and this value is called Ridge Regression or L2.



The l_2 penalty is quadratic: $\lambda \cdot \text{slope}^2$: none of the coefficients (slope) are extremely large.

The L1 penalty is the absolute value: $\lambda|\text{slope}|$: choose the most important features.

When $\lambda = 0$, the penalty is also 0, and so we are just minimizing the sum of the squared residuals. When λ asymptotically increase, we arrive to a slope close to 0: so, the larger λ is, our prediction became less sensitive to the independent variable. We can use Cross-Validation, typically 10-Fold Cross Validation is used in order to determine which λ give back the lowest VARIANCE. λ is the Tuning Parameter that controls the bias-variance tradeoff and we estimate its best value via cross-validation.

12. What is VIF? What is the suitable value of a VIF for a feature to be included in a regression modelling?

ANS: A variance inflation factor(VIF)

detects multicollinearity in regression analysis. Multicollinearity is when there's correlation between predictors (i.e. independent variables) in a model; it's presence can adversely affect your regression results. The VIF estimates how much the variance of a regression coefficient is inflated due to multicollinearity in the model.

VIFs are usually calculated by software, as part of regression analysis. You'll see a VIF column as part of the output. VIFs are calculated by taking a predictor, and regressing it against every other predictor in the model. This gives you the R-squared values, which can then be plugged into the VIF formula. "i" is the predictor you're looking at (e.g. x_1 or x_2):

$$\text{VIF} = \frac{1}{1 - R_i^2}$$

Interpreting the Variance Inflation Factor

Variance inflation factors range from 1 upwards. The numerical value for VIF tells you (in decimal form) what percentage the variance (i.e. the standard error squared) is inflated for each coefficient. For example, a VIF of 1.9 tells you that the variance of a particular coefficient is 90% bigger than what you would expect if there was no multicollinearity — if there was no correlation with other predictors.

A **rule of thumb** for interpreting the variance inflation factor:

- 1 = not correlated.
- Between 1 and 5 = moderately correlated.
- Greater than 5 = highly correlated.

Exactly how large a VIF has to be before it causes issues is a subject of debate. What is known is that the more your VIF increases, the less reliable your regression results are going to be. In general, a VIF above 10 indicates high correlation and is cause for concern. Some authors suggest a more conservative level of 2.5 or above.

Sometimes a high VIF is no cause for concern at all. For example, you can get a high VIF by including products or powers from other variables in your regression, like x and x^2 . If you have high VIFs for dummy variables representing nominal variables with three or more categories, those are usually not a problem.

13. Why do we need to scale the data before feeding it to the train the model?

ANS: Scaling of the data comes under the set of steps of data pre-processing when we are performing machine learning algorithms in the data set. As we know most of the supervised and unsupervised learning methods make decisions according to the data sets applied to them and often the algorithms calculate the distance between the data points to make better inferences out of the data.

In real life, if we take an example of purchasing apples from a bunch of apples, we go close to the shop, examine various apples and pick various apples of the same attributes. Because we have learned about the attributes of apples and we know which are better and which are not good also we know which attributes can be compromised and which can not. So if most of the apples consist of pretty similar attributes we will take less time in the selection of the apples which directly affect the time of purchasing taken by us. The moral of the example is if the apples every apple in the shop is good we will take less time to purchase or if the apples are not good enough we will take more time in the selection process which means that if the values of attributes are closer we will work faster and the chances of selecting good apples also strong.

Similarly in the machine learning algorithms if the values of the features are closer to each other there are chances for the algorithm to get trained well and faster instead of the data set where the data points or features values have high differences with each other will take more time to understand the data and the accuracy will be lower.

So if the data in any conditions has data points far from each other, scaling is a technique to make them closer to each other or in simpler words, we can say that the scaling is used for making data points generalized so that the distance between them will be lower.

As we know, most of the machine learning models learn from the data by the time the learning model maps the data points from input to output. And the distribution of the data points can be different for every feature of the data. Larger differences between the data points of input variables increase the uncertainty in the results of the model.

14. What are the different metrics which are used to check the goodness of fit in linear regression?

1) Mean Absolute Error(MAE)

MAE is a very simple metric which calculates the absolute difference between actual and predicted values.

To better understand, let's take an example you have input data and output data and use Linear Regression, which draws a best-fit line.

Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset.

so, sum all the errors and divide them by a total number of observations And this is MAE. And we aim to get a minimum MAE because this is a loss.

Advantages of MAE

- The MAE you get is in the same unit as the output variable.

The diagram illustrates the formula for Mean Absolute Error (MAE) with the following components and annotations:

- MAE =**: The metric being calculated.
- $\frac{1}{N}$** : An arrow points to this term with the label "Divide by total Number of Data Points".
- \sum** : The summation symbol.
- $|Y - \hat{Y}|$** : The absolute value of the residual. An arrow points to this term with the label "Sum Of" (referring to the summation) and another arrow points to the entire term with the label "Absolute Value of residual".
- Actual Output**: An arrow points to Y .
- Predicted Output**: An arrow points to \hat{Y} .

- It is most Robust to outliers.

Disadvantages of MAE

- The graph of MAE is not differentiable so we have to apply various optimizers like Gradient descent which can be differentiable.

```
from sklearn.metrics import mean_absolute_error  
print("MAE",mean_absolute_error(y_test,y_pred))
```

Now to overcome the disadvantage of MAE next metric came as MSE.

2) Mean Squared Error(MSE)

MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.

So, above we are finding the absolute difference and here we are finding the squared difference.

What actually the MSE represents? It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n} \sum \left(\underbrace{y - \hat{y}}_{\substack{\text{The square of the difference} \\ \text{between actual and} \\ \text{predicted}}} \right)^2$$

Advantages of MSE

The graph of MSE is differentiable, so you can easily use it as a loss function.

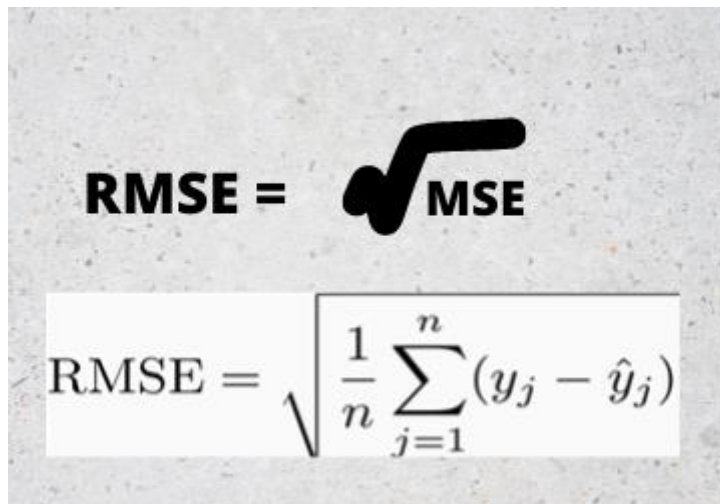
Disadvantages of MSE

- The value you get after calculating MSE is a squared unit of output. for example, the output variable is in meter(m) then after calculating MSE the output we get is in meter squared.
- If you have outliers in the dataset then it penalizes the outliers most and the calculated MSE is bigger. So, in short, It is not Robust to outliers which were an advantage in MAE.

```
from sklearn.metrics import mean_squared_error
print("MSE",mean_squared_error(y_test,y_pred))
```

3) Root Mean Squared Error(RMSE)

As RMSE is clear by the name itself, that it is a simple square root of mean squared error.



The image shows a hand-drawn diagram on a textured background. At the top, it says "RMSE = " followed by a large, bold, hand-drawn square root symbol, and then "MSE". Below this, there is a white rectangular box containing the mathematical formula for RMSE:
$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2}$$

Advantages of RMSE

- The output value you get is in the same unit as the required output variable which makes interpretation of loss easy.

Disadvantages of RMSE

- It is not that robust to outliers as compared to MAE.

for performing RMSE we have to NumPy NumPy square root function over MSE.

```
print("RMSE", np.sqrt(mean_squared_error(y_test, y_pred)))
```

Most of the time people use RMSE as an evaluation metric and mostly when you are working with deep learning techniques the most preferred metric is RMSE.

4) Root Mean Squared Log Error(RMSLE)

Taking the log of the RMSE metric slows down the scale of error. The metric is very helpful when you are developing a model without calling the inputs. In that case, the output will vary on a large scale.

To control this situation of RMSE we take the log of calculated RMSE error and resultant we get as RMSLE.

To perform RMSLE we have to use the NumPy log function over RMSE.

```
print("RMSE",np.log(np.sqrt(mean_squared_error(y_test,y_pred)))  
)
```

It is a very simple metric that is used by most of the datasets hosted for Machine Learning competitions.

5) R Squared (R²)

R² score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.

In contrast, MAE and MSE depend on the context as we have seen whereas the R² score is independent of context.

So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R² squared calculates how much regression line is better than a mean line.

Hence, R² squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit.

$$\mathbf{R^2 \text{ Squared} = 1 - \frac{SSr}{SSm}}$$

SSr = Squared sum error of regression line

SSm = Squared sum error of mean line

R2 Squared

Now, how will you interpret the R2 score? suppose If the R2 score is zero then the above regression line by mean line is equal means 1 so 1-1 is zero. So, in this case, both lines are overlapping means model performance is worst, It is not capable to take advantage of the output column.

Now the second case is when the R2 score is 1, it means when the division term is zero and it will happen when the regression line does not make any mistake, it is perfect. In the real world, it is not possible.

So we can conclude that as our regression line moves towards perfection, R2 score move towards one. And the model performance improves.

The normal case is when the R2 score is between zero and one like 0.8 which means your model is capable to explain 80 per cent of the variance of data.

```
from sklearn.metrics import r2_score
r2 = r2_score(y_test,y_pred)
```

```
print(r2)
```

6) Adjusted R Squared

The disadvantage of the R² score is while adding new features in data the R² score starts increasing or remains constant but it never decreases because It assumes that while adding more data variance of data increases.

But the problem is when we add an irrelevant feature in the dataset then at that time R² sometimes starts increasing which is incorrect.

Hence, To control this situation Adjusted R Squared came into existence.

$$R_a^2 = 1 - \left[\left(\frac{n-1}{n-k-1} \right) \times (1 - R^2) \right]$$

where:

n = number of observations

k = number of independent variables

R_a² = adjusted R²

Now as K increases by adding some features so the denominator will decrease, n-1 will remain constant. R² score will remain constant or will increase slightly so the complete answer will increase and when we subtract this from one then the resultant score will decrease. so this is the case when we add an irrelevant feature in the dataset.

And if we add a relevant feature then the R² score will increase and 1-R² will decrease heavily and the denominator will also decrease

so the complete term decreases, and on subtracting from one the score increases.

```
n=40
k=2
adj_r2_score = 1 - ((1-r2)*(n-1)/(n-k-1))
print(adj_r2_score)
```

Hence, this metric becomes one of the most important metrics to use during the evaluation of the model.

15. From the following confusion matrix calculate sensitivity, specificity, precision, recall and accuracy.

Actual/Predicted	True	False
True	1000	50
False	250	1200

ANS: A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Let's start with an example confusion matrix for a binary classifier (though it can easily be extended to the case of more than two classes):

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- In reality, 105 patients in the sample have the disease, and 60 patients do not.

Let's now define the most basic terms, which are whole numbers (not rates):

- true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.
- true negatives (TN): We predicted no, and they don't have the disease.
- false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

I've added these terms to the confusion matrix, and also added the row and column totals:

n=165		Predicted: NO	Predicted: YES	
Actual: NO		TN = 50	FP = 10	60
Actual: YES		FN = 5	TP = 100	105
		55	110	

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

- Accuracy: Overall, how often is the classifier correct?
 - $(TP+TN)/total = (100+50)/165 = 0.91$
- Misclassification Rate: Overall, how often is it wrong?
 - $(FP+FN)/total = (10+5)/165 = 0.09$
 - equivalent to 1 minus Accuracy
 - also known as "Error Rate"
- True Positive Rate: When it's actually yes, how often does it predict yes?
 - $TP/actual\ yes = 100/105 = 0.95$
 - also known as "Sensitivity" or "Recall"
- False Positive Rate: When it's actually no, how often does it predict yes?
 - $FP/actual\ no = 10/60 = 0.17$
- True Negative Rate: When it's actually no, how often does it predict no?
 - $TN/actual\ no = 50/60 = 0.83$
 - equivalent to 1 minus False Positive Rate
 - also known as "Specificity"
- Precision: When it predicts yes, how often is it correct?
 - $TP/predicted\ yes = 100/110 = 0.91$
- Prevalence: How often does the yes condition actually occur in our sample?
 - $actual\ yes/total = 105/165 = 0.64$

A couple other terms are also worth mentioning: