Rakamin Academy

HOME CREDIT

Virtual Internship Experience

Machine Learning

Model Evaluation Matrics

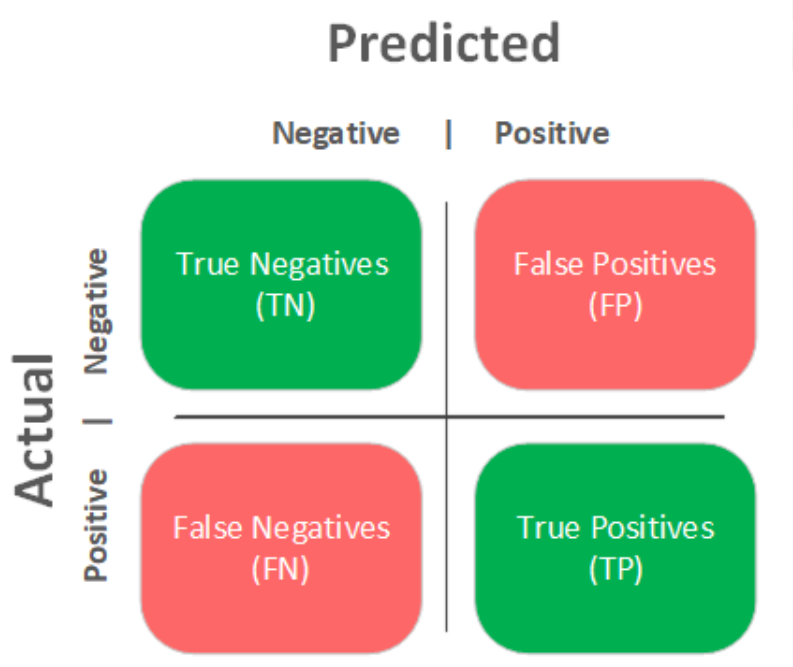
Evaluating a machine learning model.

A machine learning model can be trained extensively with many parameters and new techniques but as long as you are skipping its evaluation, you cannot trust it. so, how to evaluate your model, and practical advice for improving the model based on what we learn evaluating it. In this session, I'll discuss common metrics used to evaluate models.

A. Classification metrics

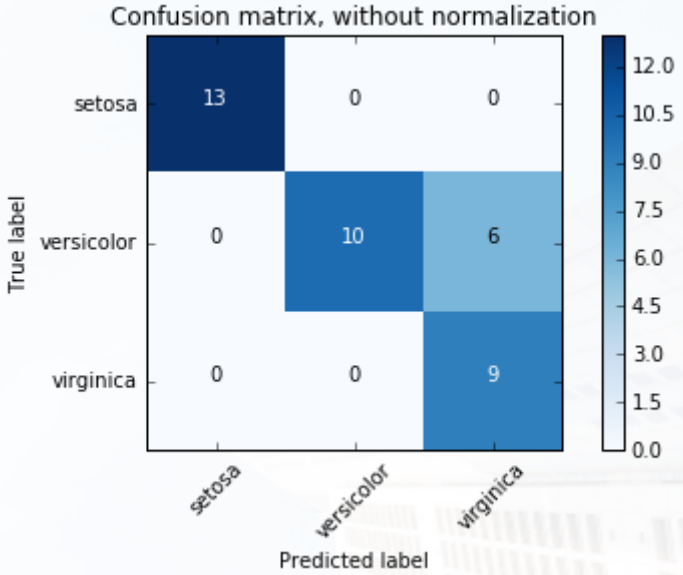
When performing classification predictions, there's four types of outcomes that could occur.

- True positives are when you predict an observation belongs to a class and it actually does belong to that class.
- True negatives are when you predict an observation does not belong to a class and it actually does not belong to that class.
- False positives occur when you predict an observation belongs to a class when in reality it does not.
- False negatives occur when you predict an observation does not belong to a class when in fact it does.



You would generate this matrix after making predictions on your test data and then identifying each prediction as one of the four possible outcomes described above.

You can also extend this confusion matrix to plot multi-class classification predictions. The following is an example confusion matrix for classifying observations from the Iris flower dataset.



The three main metrics used to evaluate a classification model are accuracy, precision, and recall.

1. **Accuracy**

   Accuracy is defined as the percentage of correct predictions for the test data. It can be calculated easily by dividing the number of correct predictions by the number of total predictions.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{all predictions}}$$

   Before going to the failure cases of accuracy, let me introduce you with two types of data sets:

   1. **Balanced**: A data set which contains almost equal entries for all labels/classes. E.g out of 1000 data points 600 are positive and 400 are negative.
   2. **Imbalanced:** A data set which contains biased distribution of entries towards a particular label/class. E.g. out of 1000 entries 990 are positive class, 10 are negative class.

Very Important: Never use accuracy as a measure when dealing with an imbalanced test set.

2. **Precision**

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class.

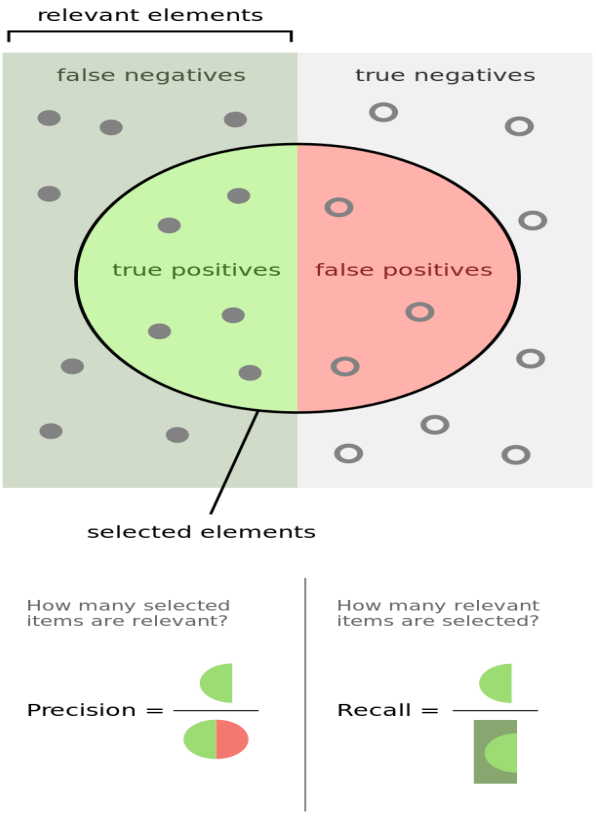
$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

3. **Recall**

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

The following graphic does a phenomenal job visualizing the difference between precision and recall.



Precision and recall are useful in cases where classes aren't evenly distributed. The common example is for developing a classification algorithm that predicts whether or not someone has a disease. If only a small percentage of the population (let's say 1%) has this disease, we could build a classifier that always predicts that the person does not have the disease, we would have built a model which is 99% accurate and 0% useful.

However, if we measured the recall of this useless predictor, it would be clear that there was something wrong with our model. In this example, recall ensures that we're not overlooking the people who have the disease, while precision ensures that we're not misclassifying too many people as having the disease when they don't. Obviously, you wouldn't want a model that incorrectly predicts a person has cancer (the person would end up in a painful and expensive treatment process for a disease they didn't have) but you also don't want to incorrectly predict a person does not have cancer when in fact they do. Thus, it's important to evaluate both the precision and recall of a model.

Ultimately, it's nice to have one number to evaluate a machine learning model just as you get a single grade on a test in school. Thus, it makes sense to combine the precision and recall metrics; the common approach for combining these metrics is known as the f-score.

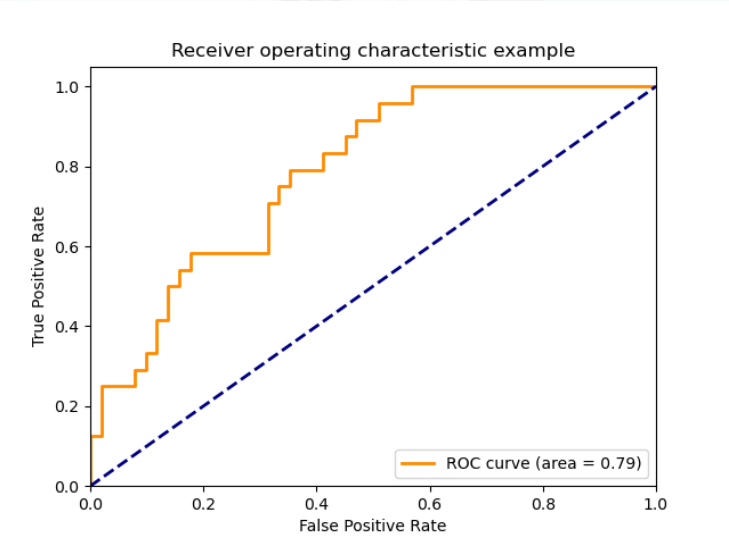
$$F_\beta = (1 + \beta^2) \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

The β parameter allows us to control the tradeoff of importance between precision and recall. β<1 focuses more on precision while β>1 focuses more on recall.

4. **ROC & AUC**

Receiver Operating Characteristic Curve (ROC):

It is a plot between TPR (True Positive Rate) and FPR (False Positive Rate) calculated by taking multiple threshold values from the reverse sorted list of probability scores given by a model.



Now, how do we plot ROC?
Just consider the M1 model which has a probability score. Then, we have assigned the data point which has a score more than 0.5 as class 1. Now sort all the values in descending order of probability scores and one by one take threshold values equal to all the probability scores. Then we will have threshold values = [0.96,0.94,0.92,0.14,0.11,0.08]. Corresponding to each threshold value to predict the classes and calculate TPR and FPR. You will get 6 pairs of TPR & FPR. Just plot them, and you will get the ROC curve.

Note: Since maximum TPR and FPR value is 1, the area under curve (AUC) of ROC lies between 0 and 1.

Area under the blue dashed line is 0.5. AUC = 0 means very poor model, AUC = 1 means perfect model. As long as your model's AUC score is more than 0.5. your model is making sense because even a random model can score 0.5 AUC.

Very Important: You can get very high AUC even in a case of dumb model generated from imbalanced data set. So always be careful while dealing with an imbalanced data set.

Note: AUC has nothing to do with the numerical values probability scores as long as order is maintained. AUC for all the models will be

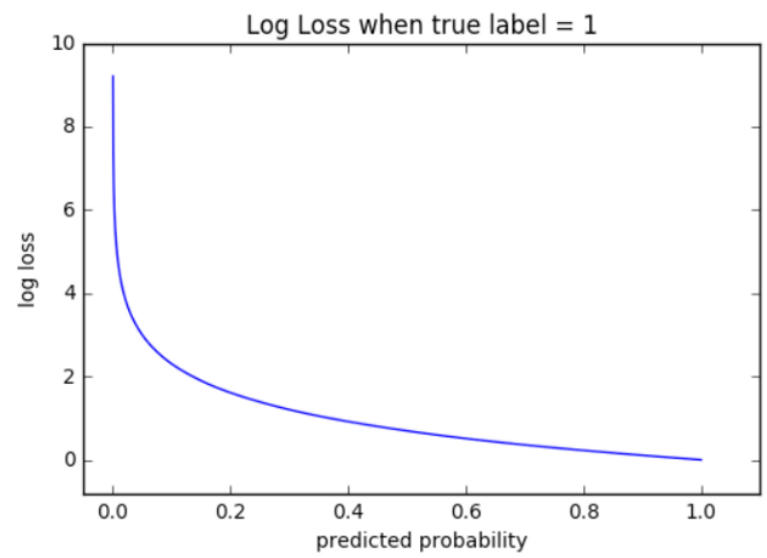
the same as long as all the models give the same order of data points after sorting based on probability scores.

5. **Log-Loss**

This performance metric checks the deviation of probability scores of the data points from the cut off score and assigns a penalty proportional to the deviation. For each data point in a binary classification, we calculate it's log loss using the formula below,

$$-(y\log(p)+(1-y)\log(1-p))$$

here p = probability of the data point to belong to class 1, and y is the class label (0 or 1). Suppose p_1 for some x_1 is 0.95 and p_2 for some x_2 is 0.55 and the cut off probability for qualifying for class 1 is 0.5. Then both qualify for class 1 but log loss of p_2 will be much more than log loss of p_1.



As you can see from the curve, the range of log loss is [0, infinity).
For each data point in a multi class classification, we calculate it's log loss using the formula below,

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$$

y(o,c)=1 if x(o,c) belongs to class 1. Rest of the concept is the same.

6. Coefficient of Determination

It is denoted by R^2.While predicting target values of the test set we encounter a few errors (e_i) which is the difference between predicted value and actual value.

Let's say we have a test set with n entries. As we know all the data points will have a target value say [y1,y2,y3…….yn]. Let us take the predicted values of the test data be [f1,f2,f3,……fn].

Calculate the Residual Sum of Squares ,which is the sum of all the errors (e_i) squared , by using this formula where fi is the predicted target value by a model for i'th data point.

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

Take the mean of all the actual target values:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

Then calculate Total Sum of Squares which is proportional to the variance of the test set target values:

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$$

If you observe both the formulas of sum of squares you can see that the only difference is the 2nd term i.e. y_bar and fi. Total sum of squares somewhat gives us an intuition that it is same as residual sum of squares only but with predicted values as [ȳ, ȳ, ȳ,…….ȳ ,n times]. Yes, your intuition is right. Let's say there is a very simple mean model which gives prediction the average of target values every time irrespective of input data.

Now we formulate R^2 as:

As you can see now, R^2 is a metric to compare your model with a very simple mean model which returns the average of target values every time irrespective of input data. The comparison has 4 cases:

case 1: SS_R = 0

(R^2 = 1) Perfect model with no errors at all.

case 2: SS_R > SS_T

(R^2 < 0) Model is even worse than the simple mean model.

case 3: SS_R = SS_T

(R^2 = 0) Model is the same as the simple mean model.

case 4:SS_R < SS_T

(0<R^2 <1) Model is okay.

B. Regression metrics

Evaluation metrics for regression models are quite different from the above metrics we discussed for classification models because we are now predicting in a continuous range instead of a discrete number of classes. If your regression model predicts the price of a house to be $400K and it sells for $405K, that's a pretty good prediction. However, in the classification examples we were only concerned with whether or not a prediction was correct or incorrect, there was no ability to say a prediction was "pretty good". Thus, we have a different set of evaluation metrics for regression models.

Explained variance compares the variance within the expected outcomes, and compares that to the variance in the error of our model. This metric essentially represents the amount of variation in the original dataset that our model is able to explain.

$$EV\left(y_{true}, y_{pred}\right) = 1 - \frac{Var\left(y_{true} - y_{pred}\right)}{y_{true}}$$

Mean squared error is simply defined as the average of squared differences between the predicted output and the true output. Squared error is commonly used because it is agnostic to whether the prediction was too high or too low, it just reports that the prediction was incorrect.

$$\text{MSE}\left(y_{true}, y_{pred}\right) = \frac{1}{n_{samples}} \sum \left(y_{true} - y_{pred}\right)^2$$

The R2 coefficient represents the proportion of variance in the outcome that our model is capable of predicting based on its features.

$$R^2\left(y_{true}, y_{pred}\right) = 1 - \frac{\sum \left(y_{true} - y_{pred}\right)^2}{\sum \left(y_{true} - \bar{y}\right)^2}$$

$$\bar{y} = \frac{1}{n_{samples}} \sum y_{true}$$

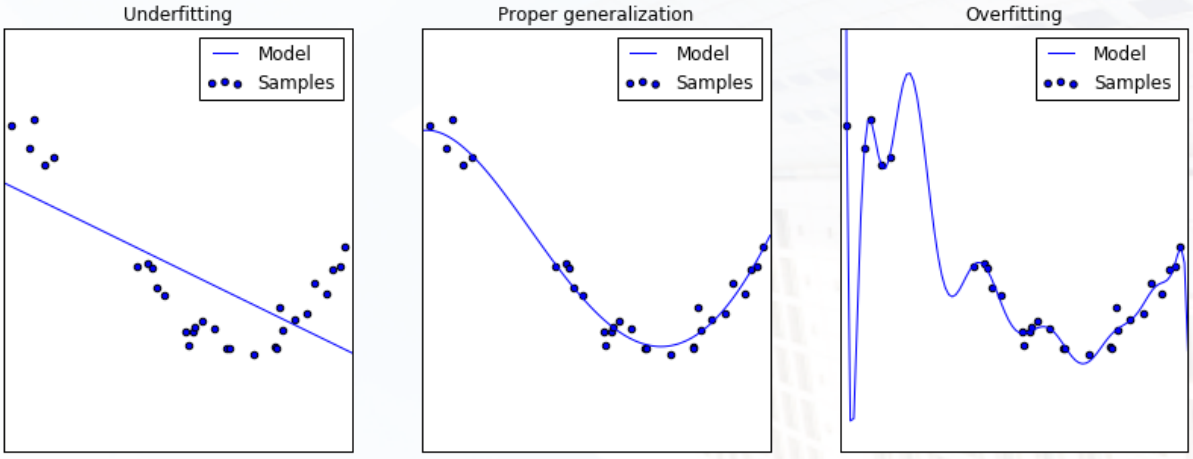
Bias vs Variance

The ultimate goal of any machine learning model is to learn from examples and generalize some degree of knowledge regarding the task we're training it to perform. Some machine learning models provide the framework for generalization by suggesting the underlying structure of that knowledge. For example, a linear regression model imposes a framework to learn linear relationships between the information we feed it. However, sometimes we provide a model with too much pre-built structure that we limit the model's ability to learn from the examples - such as the case where we train a linear model on an exponential dataset. In this case, our model is biased by the pre-imposed structure and relationships.
Models with high bias pay little attention to the data presented; this is also known as underfitting.

It's also possible to bias a model by trying to teach it to perform a task without presenting all of the necessary information. If you know the constraints of the model are not biasing the model's performance yet you're still observing signs of underfitting, it's likely that you are not using enough features to train the model.

On the other extreme, sometimes when we train our model it learns too much from the training data. That is, our model captures the noise in the data in addition to the signal. This can cause wild fluctuations in the model that does not represent the true trend; in this case, we say that the model has high variance. In this case, our model does not generalize well because it pays too much attention to the training data without consideration for generalizing to new data. In other words, we've overfit the model to the training data.

In summary, a model with high bias is limited from learning the true trend and underfits the data. A model with high variance learns too much from the training data and overfits the data. The best model sits somewhere in the middle of the two extremes.



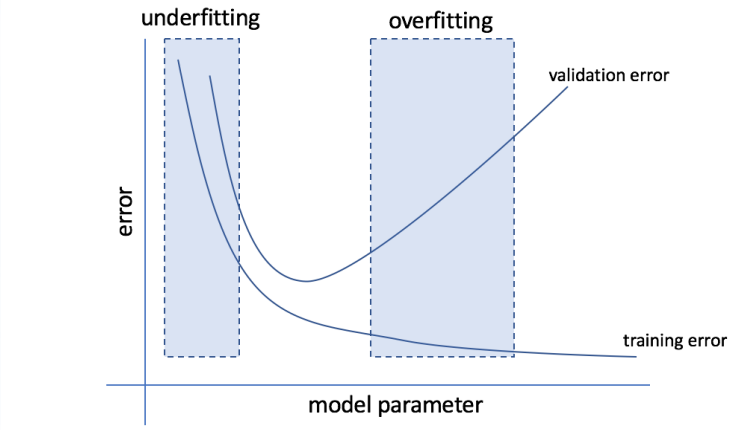
Next, I'll discuss two common tools that are used to diagnose whether a model is susceptible to high bias or variance.

Validation curves

As we discussed in the previous section, the goal with any machine learning model is generalization. Validation curves allow us to find the sweet spot between underfitting and overfitting a model to build a model that generalizes well.

A typical validation curve is a plot of the model's error as a function of some model hyperparameter which controls the model's tendency to overfit or underfit the data. The parameter you choose depends on the specific model you're evaluating; for example, you might choose to plot the degree of polynomial features (typically, this means you have polynomial features up to this degree) for a linear regression

model. Generally, the chosen parameter will have some degree of control over the model's complexity. On this curve, we plot both the training error and the validation error of the model. Using both of these errors combined, we can diagnose whether a model is suffering from high bias or high variance.

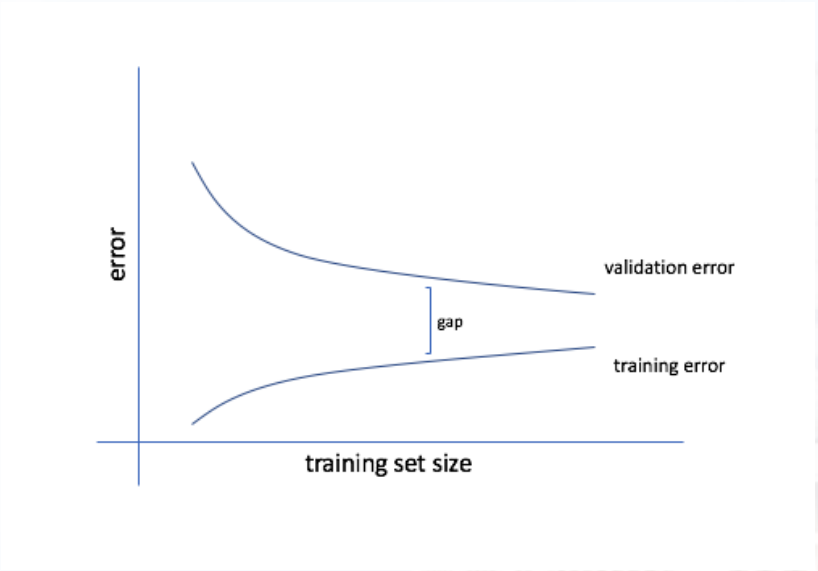


In the region where both the training error and validation error are high, the model is subject to high bias. Here, it was not able to learn from the data and it performed poorly.
In the region where the training error and validation error diverge, with the training error staying low and validation error increasing, we're beginning to see the effects of high variance. The training error is low because we're overfitting the data and learning too much from the training examples, while the validation error remains high because our model isn't able to generalize from the training data to new data.

Learning curves

The second tool we'll discuss for diagnosing bias and variance in a model is learning curves. Here, we'll plot the error of a model as a function of the number of training examples. Similar to validation curves, we'll plot the error for both the training data and validation data.
If our model has high bias, we'll observe fairly quick convergence to a high error for the validation and training datasets. If the model

suffers from high bias, training on more data will do very little to improve the model. This is because models which underfit the data pay little attention to the data, so feeding in more data will be useless. A better approach to improving models which suffer from high bias is to consider adding additional features to the dataset so that the model can be more equipped to learn the proper relationships.



If our model has high variance, we'll see a gap between the training and validation error. This is because the model is performing well for the training data, since it has been overfit to that subset, and performs poorly for the validation data since it was not able to generalize the proper relationships. In this case, feeding more data during training can help improve the model's performance.

References :

- Jordan,Yeremi, Evaluating a machine learning Model, from https://www.jeremyjordan.me/evaluating-a-machine-learning-model/ [acces:10/04/2022]
- Saurabh Raj, How to Evaluate the Performance of Your Machine Learning Model, from https://medium.com/swlh/how-to-evaluate-the-performance-of-your-machine-learning-model-40769784d654 [Acces:10/04/2022].