Model Evaluation

**CONTENTS**

1. Three datasets
2. Evaluation Criterions
   * 1. Regression
     2. Classification
3. Hyperparameter tuning

# 1. Model’s Assessment

1. For model evaluation, we create three separate sets of our dataset:
2. **Training set** – For training the model
3. **Validation set/ Dev set** – A dataset for optimising the model. The trained model will be tested on this dataset, and its hyperparameters (and other things) will be updated inorder to get high accuracy in this set.
4. **Test set** – The dataset used exclusively for testing. No improvement should be made to the model based on this set.The test set contains the examples that the learning algorithm has never seen before, so if our model performs well on predicting the labels of the examples from the test set, we say that our model generalizes well or, simply, that it’s good.

**High varience** - A model that performs well on training set, but is bad on dev set. Looks like the model is not generalising well and is overfitting the training set.

**High bias** – A model that fits both the same, but is very bad. It is underfitting both dataset.

Note that a model can have both high varience and high bias. This typically happens for higher dimensional models, where it underfits in one dimension and overfits in another.

# 2. Evaluation Criterions

## a. Regression Models

First, we compute the mean squared error (or any such error measure) for the training, and, separately, for the test data. If the MSE of the model on the test data is substantially higher than the MSE obtained on the training data, this is a sign of overfitting. If both underperforms, it is underfitting.

## b. Classification Models

#### Confusion Matrix

The confusion matrix is a table that summarizes how successful the classification model is at predicting examples belonging to various classes. One axis of the confusion matrix is the label that the model predicted, and the other axis is the actual label. Confusion matrix is used to calculate two other performance metrics: precision and recall.

It also helps us to have an idea about where the prediction is going wrong. This can help us address that problem with more data focusing on that problem.

#### Precision/Recall – Best for skewed class prediction

Precision is the ratio of **correct positive prediction**s to the **overall number of positive predictions**.

Precision = TP/ (TP+FP)

Recall is the ratio of **correct positive predictions** to the overall number of **correct predictions** in the dataset.

Recall = TP/ (TP+FN)

**High precision => if prediction == Positive: result = Positive # says + only if sure**

**High recall => if result = Positive: prediction == Positive # says + if there is a chance**

Almost always, in practice, we have to choose between a high precision or a high recall. It’s usually impossible to have both.

#### **Accuracy**

Accuracy is given by the number of correctly classified examples divided by the total number of classified examples.

**Accuracy = (TP + TN)/ (TP + TN + FP + FN)**

**Cost-sensitive Accuracy**

For dealing with the situation in which different classes have different importance, a useful metric is cost-sensitive accuracy. To compute a cost-sensitive accuracy, you first assign a cost (a positive number) to both types of mistakes: FP and FN. You then compute the counts TP, TN, FP, FN as usual and multiply the counts for FP and FN by the corresponding cost before calculating the accuracy

**Area under the ROC Curve**

didn’t read

# 3. Hyperparameter Tuning

Hyperparameters are variables that are not optimized by the learning algorithm itself. The data analyst has to “tune” hyperparameters by experimentally finding the best combination of values, one per hyperparameter.

One method is **grid search**, but trying all combinations of hyperparameters, especially if there are more than a couple of them, could be time-consuming.

There are more efficient techniques, such as random search and Bayesian hyperparameter optimization. **Random search** differs from grid search in that you no longer provide a discrete set of values to explore for each hyperparameter; instead, you provide a statistical distribution for each hyperparameter from which values are randomly sampled and set the total number of combinations you want to try. **Bayesian techniques** differ from random or grid search in that they use past evaluation results to choose the next values to evaluate.

There are also **gradient-based techniques, evolutionary optimization techniques**, and other algorithmic hyperparameter tuning techniques.