**1.1 Identify whether your algorithm suffers from a high bias (underfitting) or high variance (overfitting) problem.**

According to the given question, it is clear that the algorithm suffers from an under-fitting or high variance problem.

Justification:

Underfitting vs Overfitting:

Overfitting refers to a model that models the training data too well. Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the model ability to generalize.

Underfitting refers to a model that can neither model the training data nor generalize to new data. An underfit machine learning model is not a suitable model and will be obvious as it will has a poor performance on the training data.

To identify between underfitting and overfitting, we can check the training error and validation error of the model. We want to watch out for the cases when our training error is significantly lower than our validation error, indicating severe overfitting. The second case is underfitting, If the model is unable to reduce the training error, that could mean that our model is too simple.

**1.2 Suggest methods to address the same.**

**1) Cross Validation:**

In cross validation, instead of splitting the datasets traditionally into train and test set, we split the dataset into n-parts. We hold out or ignore any one subset (called hold out set) and train the model without the holdout set and then test the same model on the holdout set.

In k-fold cross-validation, If k=5 the dataset will be divided into 5 equal parts and the below process will run 5 times, each time with a different holdout set.

**2) Regularization:**

Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting.

Let us say that we want to build a doghouse. The model is overfitting the data and the output looks like this



This happened because of the noise present in training data. This is when regularization comes in. This is a common problem with classification problems where the model learns too much from the data including the noise or in other terms the data points that should have been ignored are caught which changes the prediction rate of the model.

Regularization significantly reduces the variance of the model, without substantial increase in its bias. It is necessary to carefully tune the parameters, as increasing the parameters could lead to underfitting.

**3) Train with more data:**

The greater the amount of noise in your data, the easier it is to overfit and the simpler model you are restricted to using. With, say, gaussian noise, increasing the amount of data in your training set increases the data-to-noise ratio, reducing overfitting. If your training and test data are from (slightly) different distributions, increasing the amount of data will do nothing to reduce this source of noise! The data-to-noise ratio will stay the same. Only other sources of noise will be eliminated.

**4) Feature Reduction:**

A model with more features tends to make a more complicated model and the more complicated model tends to be less general. There are Principal Component Analysis (PCA) and feature selection to reduce the number of features of data. The trade-off of reducing features is that you will lose the information from the data, but you will also improve the model performance.

**Question#2**

Assume that there is a total of 80 Machine learning textbooks in a library of 1000 textbooks. Suppose that a search engine retrieves 10 textbooks after a user enters ‘Machine Learning’ as a query, of which 2 are not Machine Learning textbooks. What is the precision and recall of the search engine model?

**CONFUSION MATRIX:**

* For imbalanced classification problems, the majority class is typically referred to as the negative outcome (e.g. such as “no change” or “negative test result“), and the minority class is typically referred to as the positive outcome (e.g. “change” or “positive test result”).
* The confusion matrix provides more insight into not only the performance of a predictive model, but also which classes are being predicted correctly, which incorrectly, and what type of errors are being made.
* The simplest confusion matrix is for a two-class classification problem, with negative (class 0) and positive (class 1) classes.
* In this type of confusion matrix, each cell in the table has a specific and well-understood name, summarized as follows:

|  |  |  |
| --- | --- | --- |
|  | Positive prediction | Negative prediction |
| Positive class | True positive (TP) | False positive (FP) |
| Negative class | False negative (FN) | True negative (TN) |

* The precision and recall metrics are defined in terms of the cells in the confusion matrix, specifically terms like true positives and false negatives.
* Precision is calculated as the ratio of correctly predicted positive examples divided by the total number of positive examples that were predicted.
* Precision = TruePositives / (TruePositives + FalsePositives)
* Unlike precision that only comments on the correct positive predictions out of all positive predictions, recall provides an indication of missed positive predictions.
* Recall = TruePositives / (TruePositives + FalseNegatives)

For the question, I will add the given values in the confusion matrix as a first step to calculate recall and precision

|  |  |  |
| --- | --- | --- |
|  | Actual Label (Positive) | Actual Label (Negative) |
| Predicted Label (Positive) | 8 | 2 |
| Predicted Label (Negative) | 72 | 918 |

From the table summary of confusion matrix and the given information from the question, we can learn that,

**True positive = 8**

**False negative = 72**

**False positive = 2**

**True negative = 918**

Substituting the values in formula,

**Precision = TruePositives / (TruePositives + FalsePositives)**

= 8/(8+2)

= 0.8

**Recall = TruePositives / (TruePositives + FalseNegatives)**

= 8/(8+72)

= 0.1

**Therefore, the precision and recall of the search engine model is 0.8 and 0.1 respectively.**

**Sources:**

<https://towardsdatascience.com/how-to-address-overfitting-beginner-friendly-f26d7e7dd05>

<https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e>

<https://medium.com/datadriveninvestor/how-to-handle-overfitting-and-underfitting-470a1f7389fe>

<https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85>