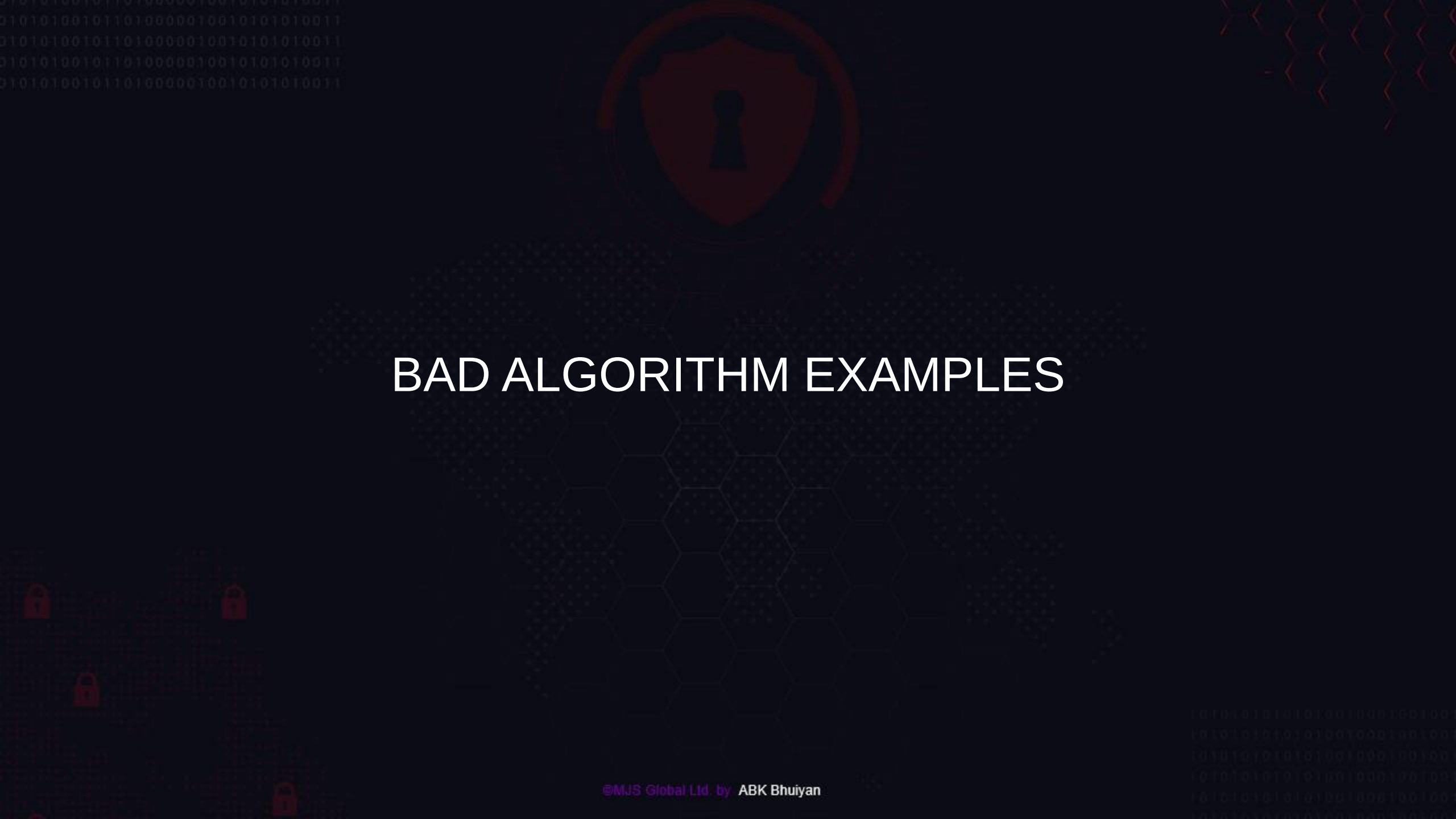
Topic - 04
Introduction to Machine Learning: Concept & Fundamentals
-- Terminologies

+88 01831-661534

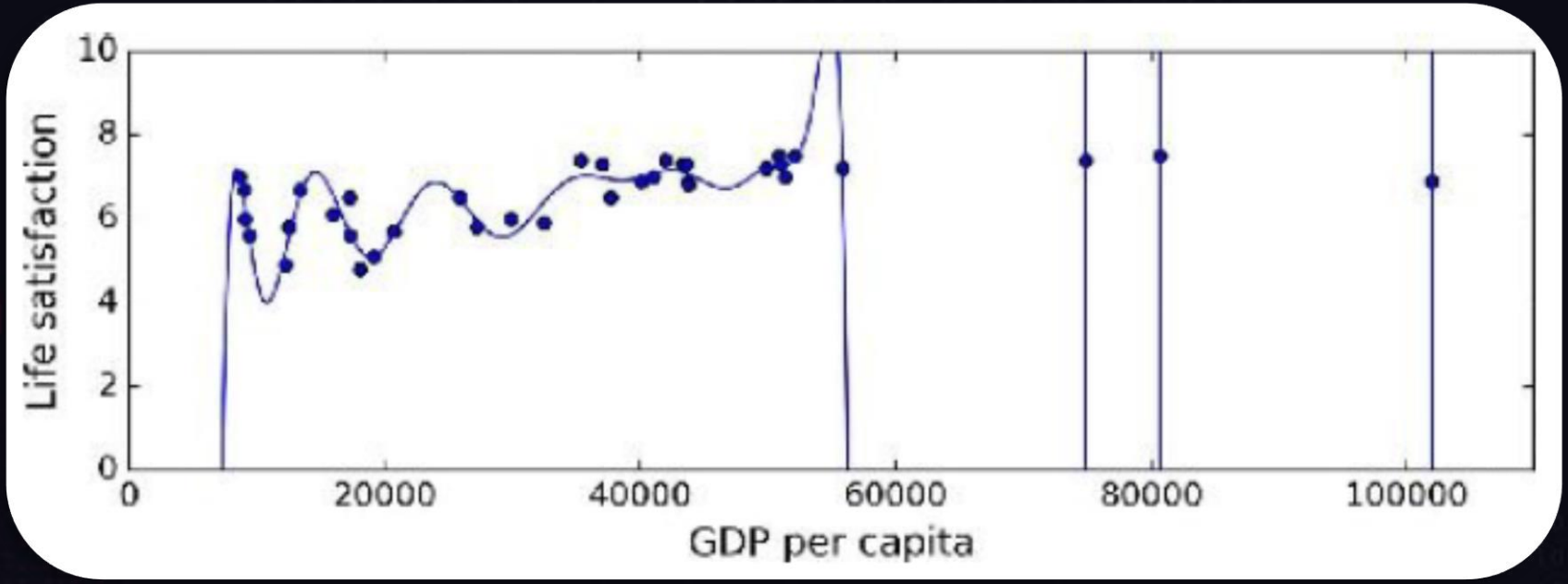
jehadfeni@gmail.com



Overfitting the Training Data

Say a foreign tourist has come to Bangladesh and the taxi driver rips the tourist off. You might be tempted to say that all taxi drivers in Bangladesh are thieves. We humans do that quite often.

Unfortunately machines can fall into the same trap if we are not careful. In Machine Learning this is called overfitting: it means that the model performs well on the training data, but it does not generalize well.



Overfitting the Training Data Cont.

In fact, Overfitting is a common incident in Machine Learning.

For example, say you feed your Life Satisfaction with Money model has many more attributes, including Country's Name. In that case, a complex model may detect patterns like the fact that all countries in the training data with a ***W*** in their name have a life satisfaction greater than 7: New Zealand (7.3), Norway (7.4), Sweden (7.2), and Switzerland (7.5).

How confident are you that the W-satisfaction rule generalizes to Rwanda or Zimbabwe? Obviously this pattern occurred in the training data by pure chance, but the model has no way to tell whether a pattern is real or simply the result of noise in the data.

How to Get Rid of Overfitting

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data.

The possible solutions are:

- To simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model)
- By reducing the number of attributes in the training data by constraining the model
- To gather more training data
- To reduce the noise in the training data (e.g., fix data errors and remove outliers)

Overfitting & Regularization

Let’s talk about another Real Life scenario where the Overfitting happens.

Let’s say you are doing Speech to Text (STT) with the help of Deep Learning.

If you keep training your data with certain features and run the process iteratively, it will also start finding patterns in the Training Data that is not needed.

For example: Let’s say there were 20 Users’ Voice Data in the Training Dataset. If we overfit the model, the Accuracy will increase but the problem that will happen is that the System will only be able to successfully do Speech to Text on those 20 Users. Here, the system is overfit with the Voice of the 20 users. It won’t work well for new Speakers.

To get rid of this, we look into Accuracy in both Training Set and Test Set and where the values are similar, we stop the Training at that stage so that the System doesn’t Overfit anymore.

Constraining a model to make it simpler and reduce the risk of overfitting is called
REGULARIZATION

Regularization Hyperparameter

The amount of regularization to apply during learning can be controlled by a Hyperparameter.

A hyperparameter is a parameter of a learning algorithm (not of the model). As such, it is not affected by the learning algorithm itself; it must be set prior to training and remains constant during training.

If you set the regularization hyperparameter to a very large value, you will get an almost flat model (a slope close to zero); the learning algorithm will almost certainly not overfit the training data, but it will be less likely to find a good solution.

Tuning hyperparameters is an important part of building a Machine Learning system

Underfitting the Training Data

Underfitting is the opposite of overfitting. It occurs when your model is too simple to learn the underlying structure of the data.

For example, a linear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.

The main options to fix this problem are:

- Selecting a more powerful model, with more parameters
- Feeding better features to the learning algorithm (Feature Engineering)
- Reducing the constraints on the model (e.g., reducing the Regularization Hyperparameter)

Testing & Validating

The only way to know how well a model will generalize to new cases is to actually try it out on new cases. One way to do that is to put our model in production and monitor how well it performs. This works well, but if our model is horribly bad, your users will complain — not the best idea.

A better option is to split our data into two sets: the Training set and the Test set. As these names imply, we train your model using the training set, and we test it using the test set. The error rate on new cases is called the ***GENERALIZATION ERROR*** (or ***OUT-OF-SAMPLE*** error),

By evaluating our model on the test set, we get an estimation of this error. This value tells us how well your model will perform on instances it has never seen before. If the ***TRAINING ERROR*** is low (i.e., your model makes few mistakes on the training set) but the generalization error is high, it means that your model is overfitting the training data.

Testing & Validating Cont.

It is common to use 80% of the data for training and hold out 20% for testing. But sometimes we also use 70%-30% or separate the data based on our needs.

Now suppose you are hesitating between two models (say a linear model and a polynomial model): which one to use? One option is to train both Models and compare how well each of them generalize using the same test set.

Now suppose that the linear model generalizes better, but you want to apply some regularization to avoid overfitting. The question is: how do you choose the value of the regularization hyperparameter?

One option is to train 100 different models using 100 different values for this hyperparameter. Let's say you find the best hyperparameter value that produces a model with the lowest generalization error - 5%. So you launch this model into production, but unfortunately it does not perform as well as expected and produces 15% errors. What went wrong?

The problem is that you measured the generalization error multiple times on the test set, and you adapted the model and hyperparameters to produce the best model for that set. This means that the model is unlikely to perform as well on new data.

Testing & Validating Cont..

A common solution to this problem is to have a second holdout set called the ***VALIDATION SET***. You train multiple models with various hyperparameters using the training set, you select the model and hyperparameters that perform best on the validation set, and when you're happy with your model you run a single final test against the test set to get an estimate of the generalization error.

To avoid "wasting" too much training data in validation sets, a common technique is to use cross-validation: the training set is split into complementary subsets, and each model is trained against a different combination of these subsets and validated against the remaining parts.

Once the model type and hyperparameters have been selected, a final model is trained using these hyperparameters on the full training set, and the generalized error is measured on the test set.

Questions to Look Into

1. How would you define Machine Learning?
2. Can you name four types of problems where it shines?
3. What is a labeled training set?
4. What are the two most common supervised tasks?
5. Can you name four common unsupervised tasks?
6. What type of Machine Learning algorithm would you use to allow a robot to walk in various
unknown terrains?
7. What type of algorithm would you use to segment your customers into multiple groups?
8. Would you frame the problem of spam detection as a supervised learning problem or an
unsupervised learning problem?
9. What is an online learning system?
10. What is out-of-core learning?

Questions to Look Into Cont.

11. What type of learning algorithm relies on a similarity measure to make predictions?
12. What is the difference between a model parameter and a learning algorithm's hyperparameter?
13. What do model-based learning algorithms search for? What is the most common strategy they use to
succeed? How do they make predictions?
14. Can you name four of the main challenges in Machine Learning?
15. If your model performs great on the training data but generalizes poorly to new instances, what is
happening? Can you name three possible solutions?
16. What is a test set and why would you want to use it?
17. What is the purpose of a validation set?
18. What can go wrong if you tune hyperparameters using the test set?
19. What is cross-validation and why would you prefer it to a validation set?