Testing Machine Learning Algorithms

In machine learning, model testing is referred to as **the process where the performance of a fully trained model is evaluated on a testing set**.

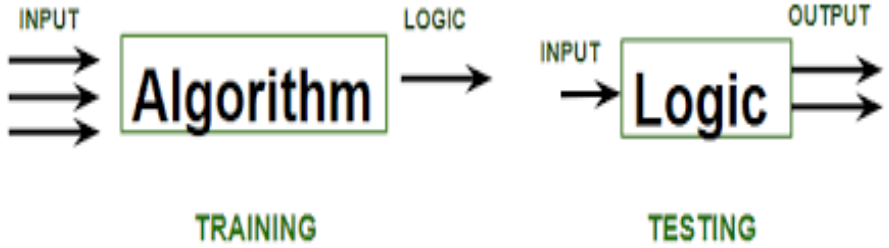
Model evaluation in machine learning testing

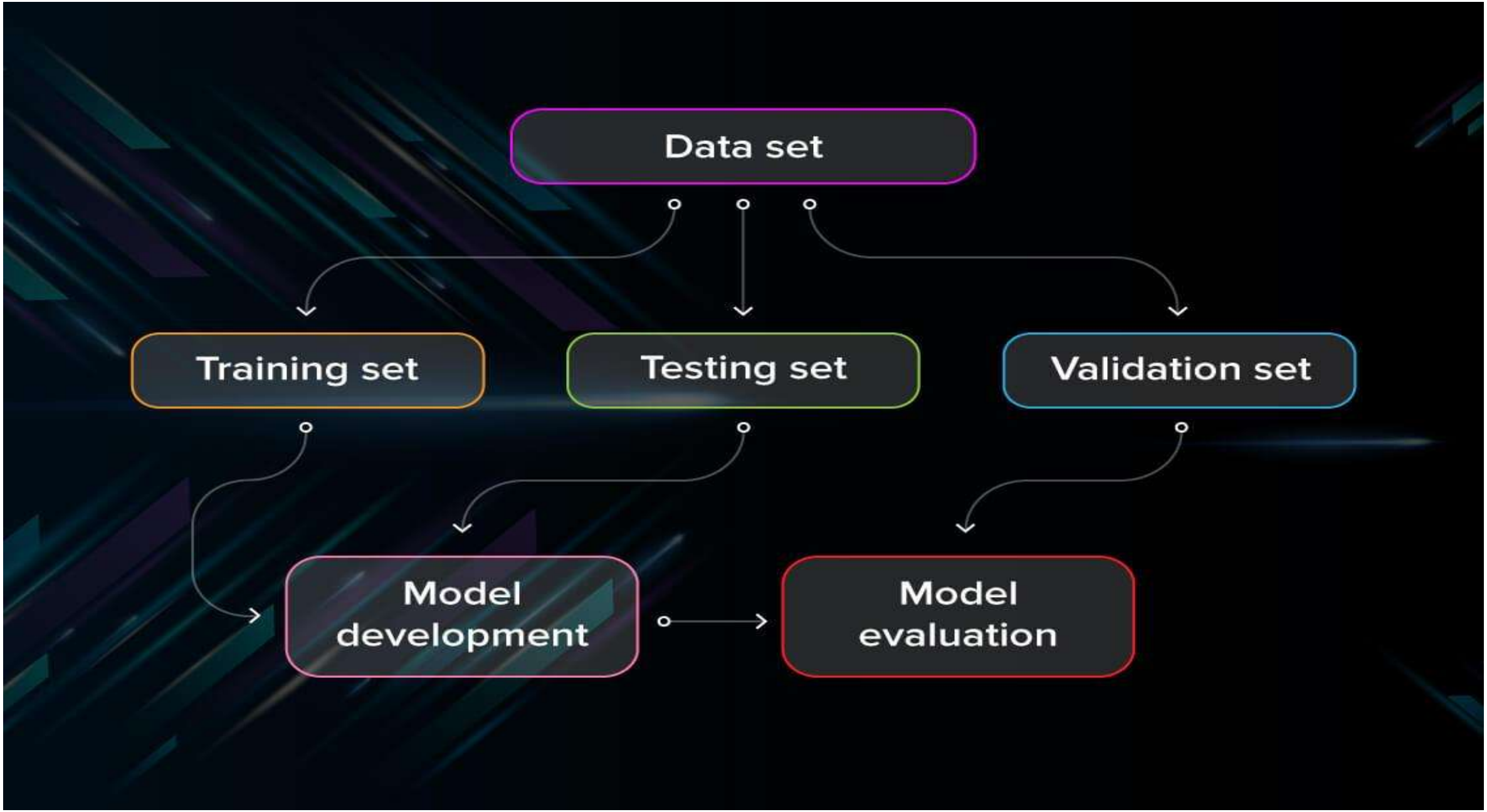
Usually, software testing includes:

•**Unit tests.** The program is broken down into blocks, and each element (unit) is tested separately. To Check the correctness of individual model components.

•**Regression tests.** They cover already tested software to see if it doesn't suddenly break. To Check whether your model breaks and test for previously encountered bugs.

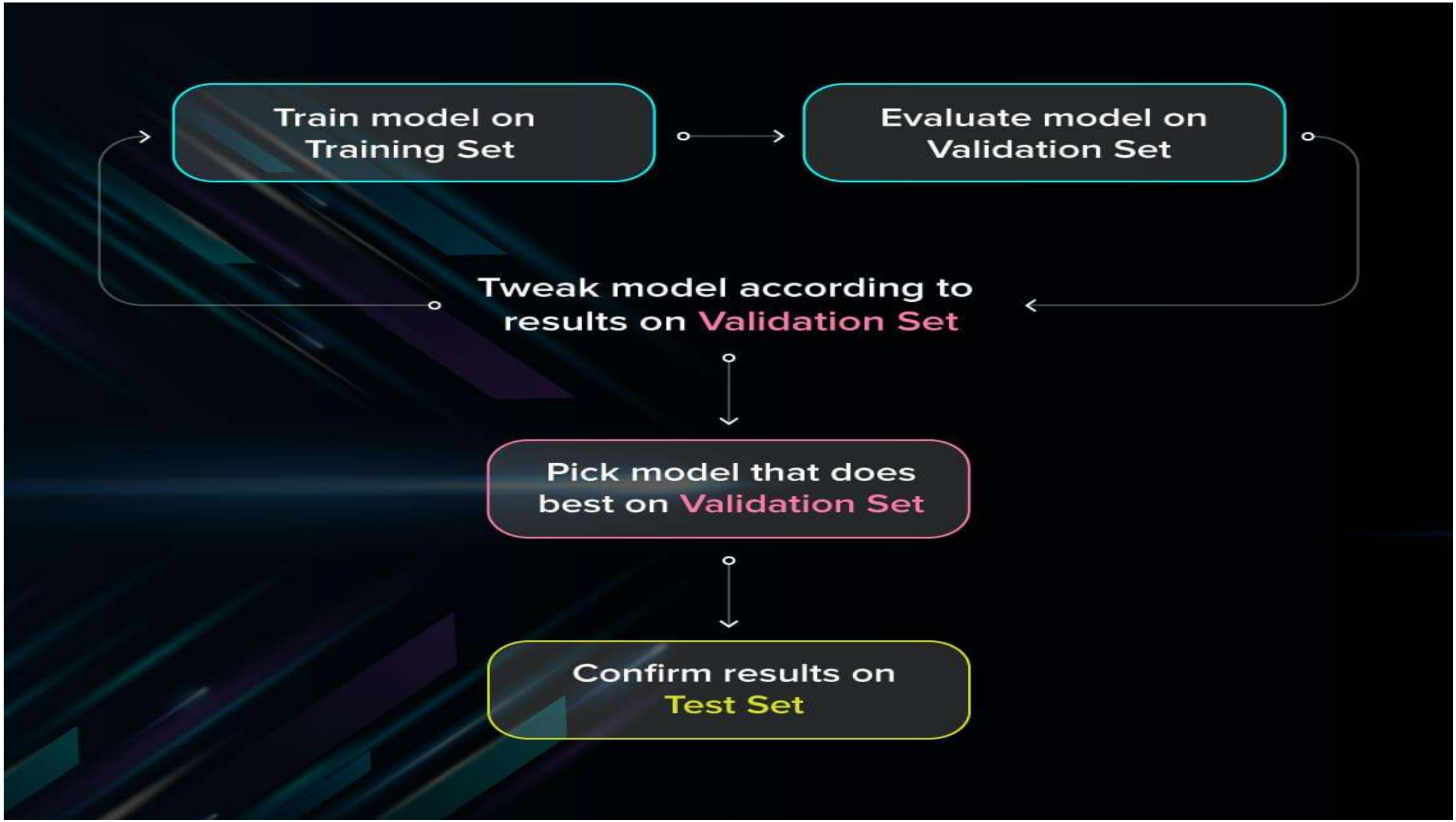
•**Integration tests.** This type of testing observes how multiple components of the program work together. To Check whether the different components work with each other within your machine learning pi




Data set
Training set
Testing set
Validation set
Model development
Model evaluation

To use a training set to train the model. Then, to evaluate the performance of the model, you use two sets of data:

- **Validation set.** Having only a training set and a testing set is not enough if you do many rounds of hyperparameter-tuning (which is always). And that can result in overfitting. To avoid that, you can select a small validation data set to evaluate a model. Only after you get maximum accuracy on the validation set, you make the testing set come into the game.
- **Test set (or holdout set).** Your model might fit the training dataset perfectly well. But where are the guarantees that it will do equally well in real-life? In order to assure that, you select samples for a testing set from your training set — examples that the machine hasn't seen before. It is important to remain unbiased during selection and draw samples at random. Also, you should not use the same set many times to avoid training on your test data. Your test set should be large enough to provide statistically meaningful results and be representative of the data set as a whole


Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set

Problems With Testing Machine Learning Models:

Software developers write code to produce deterministic behavior. Testing identifies explicitly which part of the code fails and provides a relatively coherent coverage measure. It helps us in two ways:

- Quality assurance: whether the software works according to requirements
- Identify defects and flaws during development and in production

Data Scientists and Machine Learning Engineers train models by feeding them with examples and setting parameters. The model training logic produces the behavior. This process raises the following challenges to testing machine learning models:

- **Lack of transparency**: Many models work like black boxes.
- **Indeterminate modeling outcomes**: Many models rely on stochastic algorithms and do not produce the same model after (re)training.
- **Generalizability**: Models need to work consistently in circumstances other than their training environment.
- **Unclear idea of coverage**: There is no established way to express testing coverage for machine learning models. “Coverage” does not refer to lines of code in machine learning as it does in software development. Instead, it might relate to ideas like input data and model output distribution.
- **Resource need**: Continuous testing of ML models is resource and time-intensive.

These issues make it difficult to understand the reasons behind a model’s low performance, interpret the results, and assure that our model will work even when there is a change in the input data distribution (“data drift”) or in the relationship between our input and output variables (“concept drift”).

Principles in Machine Learning Model Testing

Testing is not easy, and testing machine learning models is even harder. Need to prepare your workflow for unexpected events while working with dynamic inputs, black-box models, and shifting input/output relationships.

For this reason, some best practices in software testing:

- **Test after introducing a new component, model, or data, and after model retraining**
- **Test before deployment and production**
- **Write tests to avoid recognized bugs in the future**

Testing machine learning models has additional requirements. You also need to some testing principles specific to machine learning problems:

Robustness Interpretability Reproducibility

Robustness

Robustness requires your model to produce a relatively stable performance even in the case of radical real-time change of data and relationships.

- You can strengthen robustness in the following ways:
- Have a machine learning procedure that your team follows.
- Explicitly test for robustness (e.g., drift, noise, bias).
- Have a monitoring policy for deployed models.

Interpretability

Maintaining interpretability makes you understand specific aspects of your model:

- Whether the model predicts outputs as it should (e.g., based on human evaluators)
- How input variables contribute to the output
- Whether the data/model has underlying biases

Reproducibility

To understand how your model changes thanks to parameter adjustments, retraining, or new data, especially within a team, you need to make your results reproducible.

- Reproducibility has many aspects. Here are some tips:
- Use a fixed random seed by a deterministic random number generator.
- Make sure that the components run in the same order and receive the same random seed.
- Use version control even for preliminary iterations.