Ministry of Education and Science of Ukraine

ODESA NATIONAL UNIVERSITY OF TECHNOLOGY

International Competition of
Student Scientific Works

BLACK SEA SCIENCE 2023

PROCEEDINGS

ODESA, ONUT 2023

Ministry of Education and Science of Ukraine

Odesa National University of Technology

International Competition of Student Scientific Works

BLACK SEA SCIENCE 2023

Proceedings

Odesa, ONUT
2023

STUDY OF THE OPTIMAL STRATEGY FOR TESTING COMPLEX ML MODELS AND AI APPLICATIONS USING THE MLOPS CONCEPT

Author: Zmiivskyi Volodymyr
Advisor: Kuznetsova Yuliia
National Aerospace University
named after N. Ye. Zhukovsky
"Kharkiv Aviation Institute" (Ukraine)

Abstract: *In this paper, a study was made of the types of testing for software products using machine learning and intelligence technologies and their structuring. An analytical study was carried out to highlight the main shortcomings and ways to solve them. The entire process of software testing was analyzed, its types, stages and processes in the classical form were highlighted. After – there were recommendations for testing machine starting systems with the involvement of the MLOps methodology. Testing tools were analyzed and recommendations for testing ML models were provided. Knowledge of machine learning and its role in the development of modern IT technologies was structured.*

Keywords: *Software testing, Test strategy, ML models, AI-application, Test plan, MLOps, Types of testing.*

I. INTRODUCTION

Formulation of the problem. The world of modern IT technologies is constantly evolving. Every day there are new technologies that help humanity to live easier and better. We have come close to a new branch of the evolution of information technology – we can produce robotics, create the latest web applications, expand the boundaries of engineering. One of the manifestations of such a sustainable development of information technologies is artificial intelligence and machine learning.

Artificial intelligence is the ability of a system to correctly interpret external data, learn from such data, and use the knowledge gained to achieve specific goals and objectives through flexible adaptation. Now the use of artificial intelligence is highly valued, as a result, the demand for developers in this software area is growing rapidly. The main activities for the development of AI in Ukraine today is the construction of intelligent computer systems with integral intellectual behavior, an important property of which is adaptability to environmental changes. Developments are underway in the field of speech and visual recognition, intelligent robotics and modern telecommunication systems and mobile communication systems, systems for remote protection of objects.

Machine learning (ML) is a sub-branch of artificial intelligence in information technology that often uses statistical techniques to enable computers to "learn" (i.e., incrementally improve performance on a particular task) from data, without having to be explicitly programmable. Now this is a new technology that requires detailed study.

The essence of ML. Machine learning tasks tend to fall into two broad categories, depending on whether a training "signal" or "feedback" is available to the learner of the system.

There are many applications of machine learning today. For example, technology helps:

– to recognize the language of virtual assistants;
– to recognize handwritten letters;
– to define languages;
– to offer recommendations on websites;
– to look for documents
– to identify suspicious transactions;
– to predict the value of currencies;
– to analyze demand;
– to learn intelligent technique.

Machine learning technology is developing many different industries. Among the areas where it is used are business, medicine, the financial industry, industry and the IT sector. Like any software, ML systems and software developments for the use of artificial intelligence require testing to ensure the quality of the software product.

“Quality is the internal certainty of an object, which constitutes the specificity that distinguishes it from all others” – so it is noted in the explanatory dictionary of the Ukrainian language. In 1979 Crosby defined quality as "conformance to requirements", while Juran and Gryna in 1970 defined quality as "fitness for use". The giants of the IT industry, in pursuit of market leadership, pay special attention to quality, so software testing is an integral part of the software development life cycle. Considering how new and uncommon development area ML is, the demand for specialists of this level is very high [1].

The purpose of software testing, regardless of who tests it or where it is done, is to ensure the required quality of the software and to reduce the risk of program bugs. Therefore, the relevance of this work is to highlight the possible methods and techniques of software testing, to choose the most appropriate approach that can ensure the high quality of the software being developed.

In the structures of ML systems, it is difficult to single out the main "points" of testing and verifying a property, which is due to the huge data set that serves to develop the algorithmic grid of the entire system. The flexible methodology of MLOps helps to solve this problem [1].

MLOps – Machine Learning Operations, or DevOps for Machine Learning, is the bringing together of people, processes, and platforms to bring machine learning to business value. They simplify development and deployment by monitoring, validating, and managing machine learning models. The MLOps methodology is quite flexible, which makes it possible to test complex systems during their deployment and work with large databases [2].

Thus, there is a need to develop a software testing strategy, since by the current time there is no clearly defined testing plan for ML systems and applications using artificial intelligence technology.

The aim of the project is to analyse and research various of techniques and methods for testing ML models, as well as an experimental study of testing strategies for ML systems and AI applications to highlight the main shortcomings and ways to solve them.

To achieve the goals set, it is necessary to solve the following tasks:

– to analyze the features and possible risks when testing AI developments and ML models;

– to consider alternative software development testing methods;

– to plan a pilot study to evaluate the effectiveness of testing;

– to develop practical recommendations for solving the problem of software testing in the field of machine learning.

The scientific novelty lies in the improvement of the testing model for ML models and AI applications, which will be clear to all business units, through the use of the MLOps concept, which allows high-quality testing of complex ML systems.

Practical significance of the obtained results lies in the fact that the testing approach obtained as a result of the experiment can be used in the software development process, guaranteeing a high level of quality at the entire stage of the software development life cycle, namely in complex ML software models.

Thus, as a result of the pilot study, an actual test strategy for testing ML models and AI applications using the flexible MLOps concept will be formed, which, in turn, will allow getting rid of the so-called "technical debt" in AI applications.

II. ANALYTICAL REVIEW OF LITERATURE

Until recently, we were dealing with sizable amounts of data and very few models on a small scale. Now the situation is changing and we have to implement decision-making automation in the widest range of applications. Technological solutions in the form of developments on the use of artificial intelligence help in this. And of course, this goes hand in hand with a host of technical challenges that come with building and deploying machine learning systems.

Machine learning refers to artificial intelligence methods that teach a computer to independently solve different problems. Computers perform analytical work and identify patterns much faster than humans thanks to preloaded data and special algorithms.

Algorithms are determined depending on what problem needs to be solved and what data the developers have. A set of training data provides algorithms that process various queries with their help [2].

As a rule, computers require a large amount of information and statistics in order to learn how to make correct and necessary predictions.

In order to better understand the essence of machine learning, it should be understood that ML, like any other software area, consists of several basic equivalents (Fig. 1).

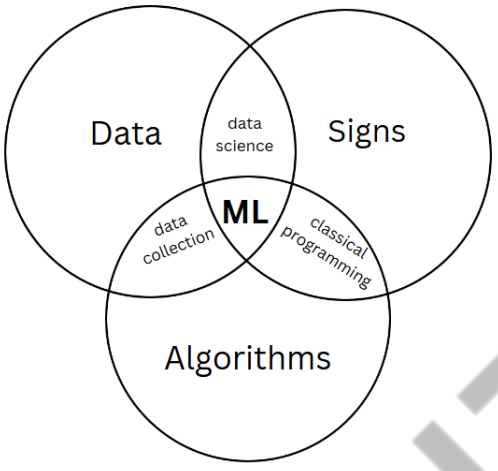

Fig. 1. Structural parts of machine learning

The first is data. In machine learning, data collection plays one of the most important roles. To train a single ML model, tens of thousands of examples are needed for the model to be able to provide a more accurate result. Of course, we must understand that checking and testing such a database is one of the tester's entry points.

The second is signs. In machine learning, signs, properties, characteristics are called "features". When training a model, it should be understood that the algorithm should pay attention only to what it needs. The more properties the algorithm knows, the slower the model runs. What to pay attention to in the end should be decided by the model itself, even if it often takes more time than training, but the initial data and possible parameters are given to it by a person, and a tester can also do this, even if it formally refers to data collection. Based on the input parameters, the algorithm will not pay attention, but we must provide the maximum picture of what we see with the input data.

The third is the algorithm. This is where developers work and write instructions for the machine learning model to work.

There are a lot of examples of modern use of ML. Machine learning is used to optimize search engines, improve work in the areas of service, medicine, and construction.

Like any software, ML modules need to be tested.

Generally speaking, testing is a technical research process designed to discover information about the quality of a product in relation to the context in which it should be used. Testing technique also includes both the process of finding bugs or other defects and testing software components for evaluation [3].

In software engineering, there are five main types of tests used at different stages of the development cycle (Fig. 2)

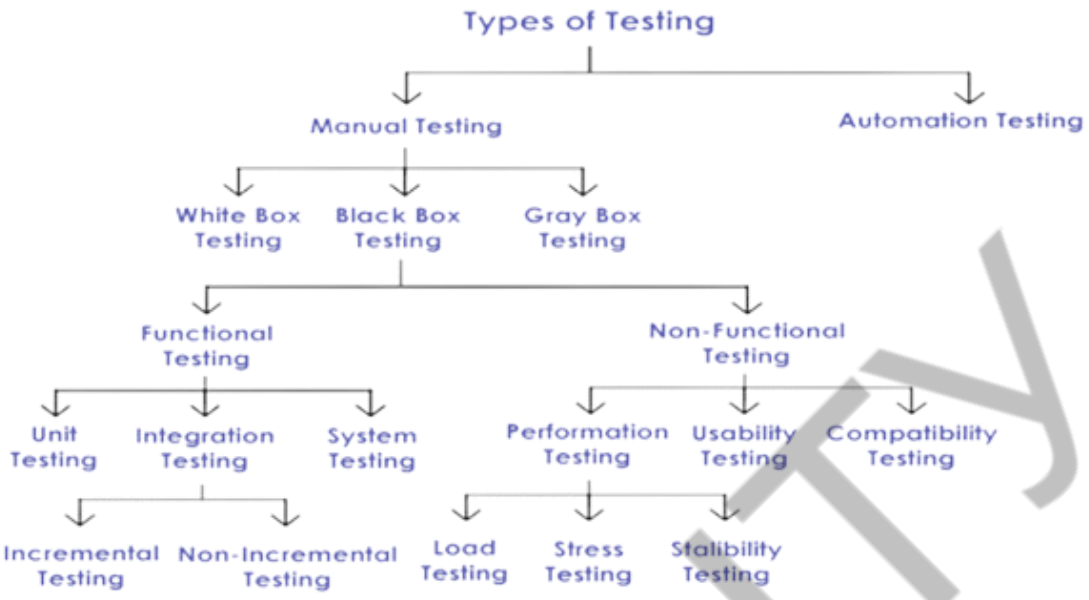

Fig. 2. Types of tests [4]

Unit tests for individual components, each with one area of responsibility and one end result, such as a list filtering function.

Integration tests for the combined verification of the combined operation of several individual components, such as data processing [4].

System tests to verify the design of the system as a whole by comparing expected results with input data;

Acceptance tests to verify that the requirements specified in the acceptance conditions are met are commonly referred to as user acceptance tests (UAT – User Acceptance Testing);

Regression tests to test previously known bugs to make sure new changes don't cause them again.

To structure and develop tests, it is recommended to use the 3A (Arrange-Act-Assert) approach, which orders inputs, actions, and results:

- Arrange – set up all kinds of inputs for testing;
- Act – apply the input data to the component under test;
- Assert – confirm the receipt of the expected result.

It should be understood that testing can be modified depending on which software area is being tested. With all the demand for machine learning, customers and executors of such projects often miss the field of testing in its classical sense. If insufficient attention is paid to testing, this can lead to incorrect operation of ML systems and cause customer disappointment in the technology as such. Of course, Data Scientists themselves check their models using special metrics, but the reality is that the service is not always able to take into account various nuances and production bottlenecks. Unfortunately, there are no well-established practices for testing ML services in the IT industry yet, therefore, when

selecting tests, we rely on our own expertise and experience of specialists, who know the industry specifics of the customer company. In classical testing, we can predict the result: how the system will react to certain input data. If we are talking about the use of ML in production environments, this predictable result can be compliance with regulatory documents: GOSTs, instructions, permanent and temporary technological maps that determine production processes and the quality of the final product. Testing allows you to make sure that the forecasts of the ML system do not contradict the regulations adopted by the enterprise. instructions, permanent and temporary technological maps that determine the production processes and the quality of the final product. Testing allows you to make sure that the forecasts of the ML system do not contradict the regulations adopted by the enterprise. instructions, permanent and temporary technological maps that determine the production processes and the quality of the final product. Testing allows you to make sure that the forecasts of the ML system do not contradict the regulations adopted by the enterprise.

One of the "outputs" in testing machine learning models of varying complexity is MLOps (Fig. 3).

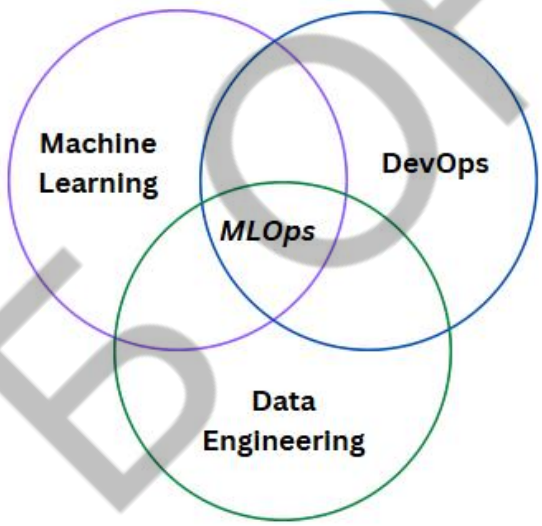

Fig. 3. The structure of the MLOps methodology

To better understand MLOps, we must first take a look at the life cycle of machine learning systems. This life cycle involves several development teams at once, covering different aspects of working with data.

In order to connect to the process, the following commands take part:

- Business development team or software product development team – defining business goals using key performance indicators (KPIs).
- Data engineers – data collection and preparation.
- Data science – designing machine learning solutions and developing models.
- IT or DevOps – deployment and monitoring with the data science team.

Managing such systems on a large scale is not an easy task. There are many bottlenecks to consider. The following are the main challenges that teams face:

- Lack of data scientists capable of developing and deploying scalable web applications. A new profile of machine learning engineers is emerging in the market these days to meet this need. Their business stands at the intersection of data science and DevOps.
- Change business goals in the model. Given the many dependencies on ever-changing data, the need to maintain model performance standards, and to provide AI-driven control, retraining a model in response to changes in business goals is not an easy task.
- Mutual misunderstanding between technical departments and business teams, which find it quite difficult to find a common language within the framework of joint work. More often it is this misunderstanding that causes the failure of large projects.
- Risk assessment. The nature of the black box of such machine learning and deep learning systems is a matter of constant debate. Models tend to deviate from what they were originally intended to do. Assessing the risk/cost of such deviations is a very important and careful step [5].

III. OBJECT, SUBJECT, AND METHODS OF RESEARCH

The object of study is use of testing methods and strategies, namely MLOps testing methods, for comparative analysis of testing strategies

The subject of study is software models and implementations of machine learning.

The study methods. This analytical study presents methods and strategies for software testing. Comparison of these types and application to developments using learning art and artificial intelligence.

When testing complex systems and applications, you should remember the main specifics of testing a software product.

Testing processes determine whether the development products of a particular activity meet the requirements of that activity and whether the system and/or software satisfies the intended use and user needs. The objectives of the testing process are defined for different levels of integrity.

These process objectives define the breadth and depth of test documentation. You can then select documentation items for each type of test documentation.

The scope of testing includes software systems, computer software, hardware and their interfaces. This standard applies to software systems being developed, maintained, or used (legacy, COTS, undeveloped items).

Any testing has its artifacts. The first is a test plan.

A Test Plan is a document that describes the entire scope of testing work, starting with a description of the object, strategy, schedule, criteria for starting and ending testing, to the equipment required in the process, special knowledge, and risk assessment with options for resolving them.

Any methodology or process tries to impose its test plan design formats. If the standard template doesn't fit, or you need to come up with your own more specific document format, then you might be told that a good test plan should describe the following:

1) functionality to be tested - description of the test object: systems, applications, equipment;
2) functionality not subject to testing - a list of functions and a description of the system to be tested and its components separately;
3) testing strategy, namely: types of testing and their application in relation to the test object;
4) the sequence of work: preparation (Test Preparation), testing (Testing), analysis of the results (Test Result Analisys) in the context of the planned development phases.
5) criteria for starting testing:
- readiness of the test platform (test board);
- completeness of the development of the required functionality;
- availability of all required documentation.

6) test completion criteria:
- test results meet the product quality criteria;
- requirements for the number of open bugs are met;
- exposure of a certain period without changing the source Code Freeze (CF);
- exposure of a certain period without opening new Zero Bug Bounce (ZBB) bugs.

All types of software testing, depending on the goals pursued, can be divided into the following groups:

1) functional;
2) non-functional;
3) associated with changes;

Functional tests are based on functions and features, as well as interactions with other systems, and can be presented at all levels of testing: component or unit (Component / Unit testing), integration (Integration testing), system (System testing) and acceptance testing (Acceptance testing). Functional types of testing consider the external behavior of the system. The following are some of the most common types of functional tests:

- Functional testing (Functional testing);
- Security and Access Control Testing;
- Interoperability Testing;

Non-functional testing describes the tests necessary to determine the characteristics of the software, which can be measured by different quantities. In general, this is testing how the system works [6].

The main types of non-functional tests:

1) All types of performance testing:

- load testing (Performance and Load Testing);
- stress testing (Stress Testing);
- stability or reliability testing (Stability/Reliability Testing);
- volume testing (Volume Testing);

2) Installation testing;
3) Usability Testing;
4) Testing for failure and recovery (Failover and Recovery Testing);
5) Configuration Testing;
6) Security and Access Control Testing;
7) Localization and internationalization testing.

After making necessary changes, such as fixing a bug/defect, the software should be tested again to confirm that the problem has indeed been fixed. The following are the types of testing that should be performed after software deployment to confirm that the application is working or that a defect has been corrected:

- smoke testing (Smoke Testing);
- regression testing (Regression Testing);
- assembly testing (Build Verification Test);
- sanitary testing or consistency / serviceability testing (Sanity Testing).

Currently, there are many types of testing, there are also a large number of classifications of these types. The main classification of types of testing occurs according to program goals. The figure below shows the classification of testing types (Fig. 4).

Also, it should be understood that testing has a level structure. Testing at different levels is carried out throughout the life cycle of software development and maintenance.

The level of testing determines what tests are performed on: on a single module, a group of modules, or the system as a whole. Testing at all levels of the system is the key to successful implementation and delivery of the project [7].

Software testing levels describe the stages of software development when testing is carried out. Generally, there are four progressive levels of testing based on the area they focus on in the software development process: unit testing, integration testing, system testing, and user acceptance testing (UAT).

Risks arising from software testing:

PMBOK recommends managing risk in 4 steps:

1) Identification. Identify risks that may interfere with project objectives.
2) Analysis. Determine which of the identified risks are the most dangerous.
3) Layout. Plan for dangerous risks.
4) Monitoring and control. Keep the project plan and risk list up to date [8].

A distinctive feature of software testing using ML technology. In order to understand the differences, you need to understand how the system works. Is there any difference between classical algorithms/hardcoded logic and functionality based on ML models?

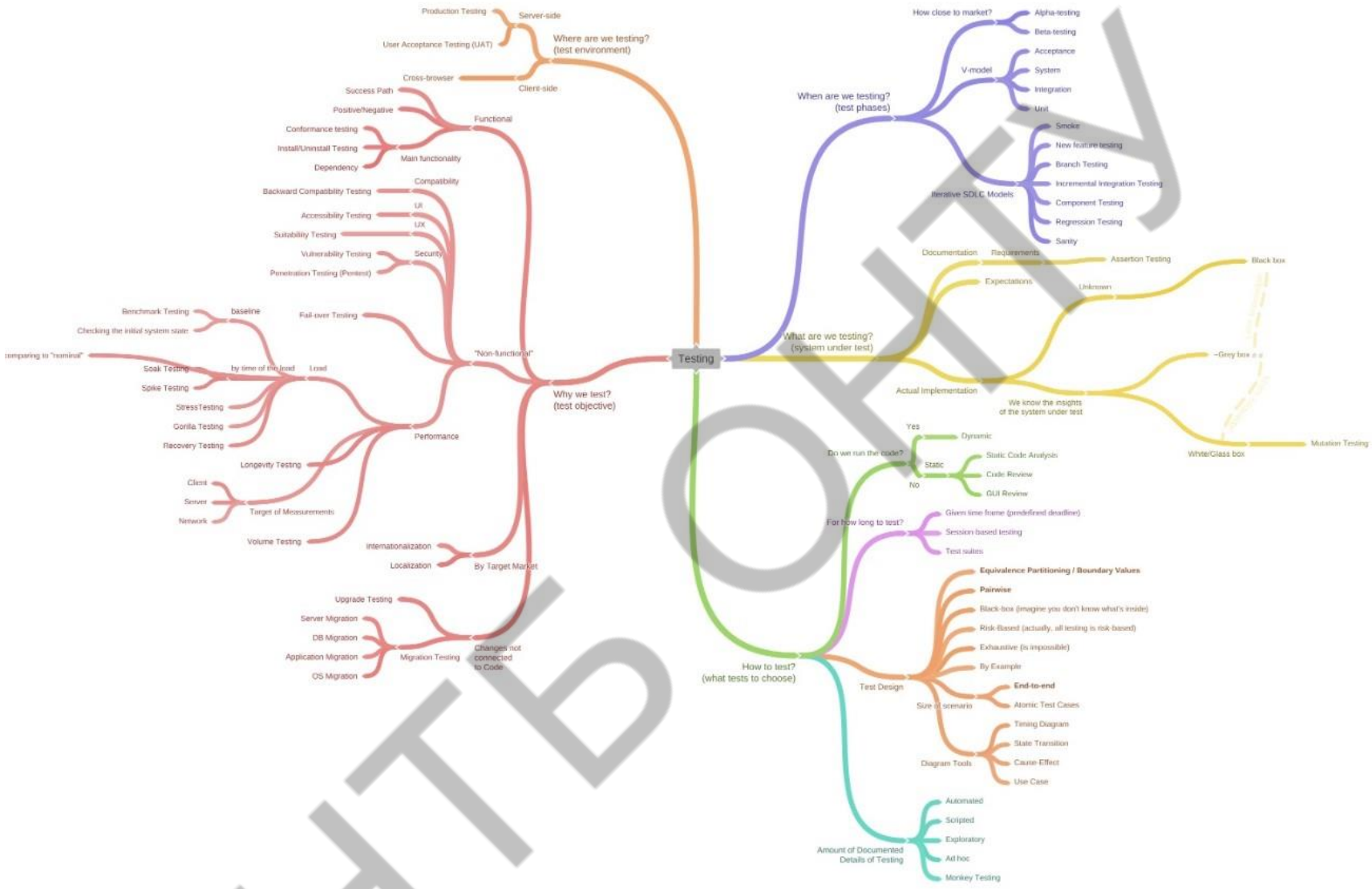

Fig. 4. Typical branching of the classification of types of testing [6]

From the point of view of the blackbox approach, this is still the same box with inputs and outputs.

However, inside the box, namely how the system is built, is somewhat different. The main difference is as follows:

When writing functions, we are guided by the rules that we set ourselves (traditional programming) or the function with its coefficients is selected by the algorithm based on the data we give it (machine learning).

Typical development process:

- Task formulation based on business requirements;
- Collection, cleaning and preparation of data (because garbage in – garbage out);
- selection of alternative learning algorithms based on the type of task, data availability and quality;
- Algorithm parameterization;
- Splitting the data set into training and test samples;
- Launching the training procedure on the required infrastructure;
- Iterative training with the required number of epochs, with regular testing on isolated test data samples;
- Preparing the model for production use (often involves reimplementation on another technology to optimize performance).
- Integration of model-based functionality in the production system.

Traditional QA occurs at the stage when the model has already been trained and it is possible to integrate it into the system:

1. Test cases have different test datasets. When testing, you should extract several data sets for this purpose. At the same time, the input data for the sample is not equivalent in importance: the voice assistant recognition function should work well for “call a child”, “what is the weather now”, and is rather less demanding in recognizing combinations such as “magneplanar headphone impedance”. Accordingly, to optimize the testing time, the data can and should be ranked.

2. If we have an ensemble of models that works well on a clustered sample, we need to cover each part of the aggregate model with tests. An example is credit scoring, where one model works for households with low total income and another for high-income families. The reason is simple: people's behavior is very different, different models better describe the end result.

3. If automation tools are used to implement regression control, the specified required level of acceptable deviation of the result, as well as the permissible level of outliers. Otherwise, the tests will regularly fail. Use metrics appropriate for the task type and conduct a mandatory data scientist review for test scenarios.

4. Be sure to check if the deployed ML function processes the data correctly (+/- inversion is common). The white-box approach works well here: use tests to check the correctness of loading the input data, check the feature outputs. The more complex the data preparation process and the more automated, the more closely you need to track it.

5. Data can mutate. This means that given the same input, a different output is now expected, such as user behavior has changed. A technically well-tested model will not become old over time, but it may not be usable.

6. Public beta - test - it's just a must for ml-based systems. The data used in both training and testing, the better your model will get. Important: The data should not be extrapolated. The more diverse the test data set, the more accurate the model testing process will be.

7. Pay attention to integration testing.

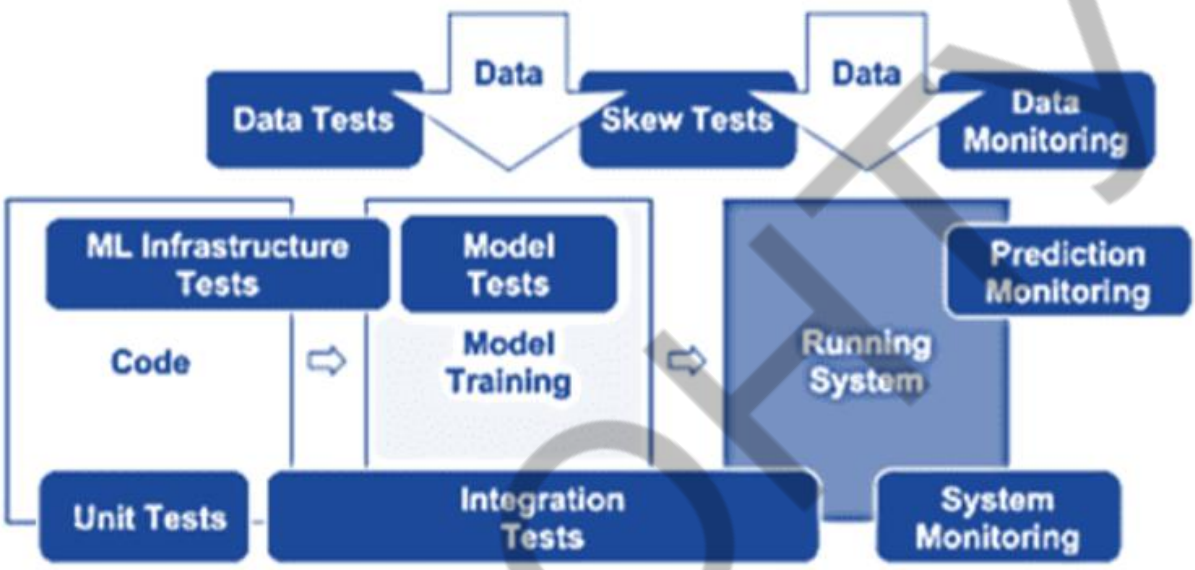

Fig. 5. Test objects in ML-system [10]

MLOps concepttries to take into account the specifics of the development, testing, deployment and operation of machine learning systems, integrating it with the best practices of software engineering.

The main task of the MLOps team is to automate the deployment of ML models in the main software system or as a service component [10].

Shown below (Fig. 6) is the end-to-end reference design for MLOps. It is important that the life cycle of MLOps is an iterative, not a linear process.

A failed test or compilation issue are examples of conditions for reverting to an earlier DevOps stage. MLOps inherits DevOps terms and adds new ones, such as offline validation and model drift, unique to machine learning.

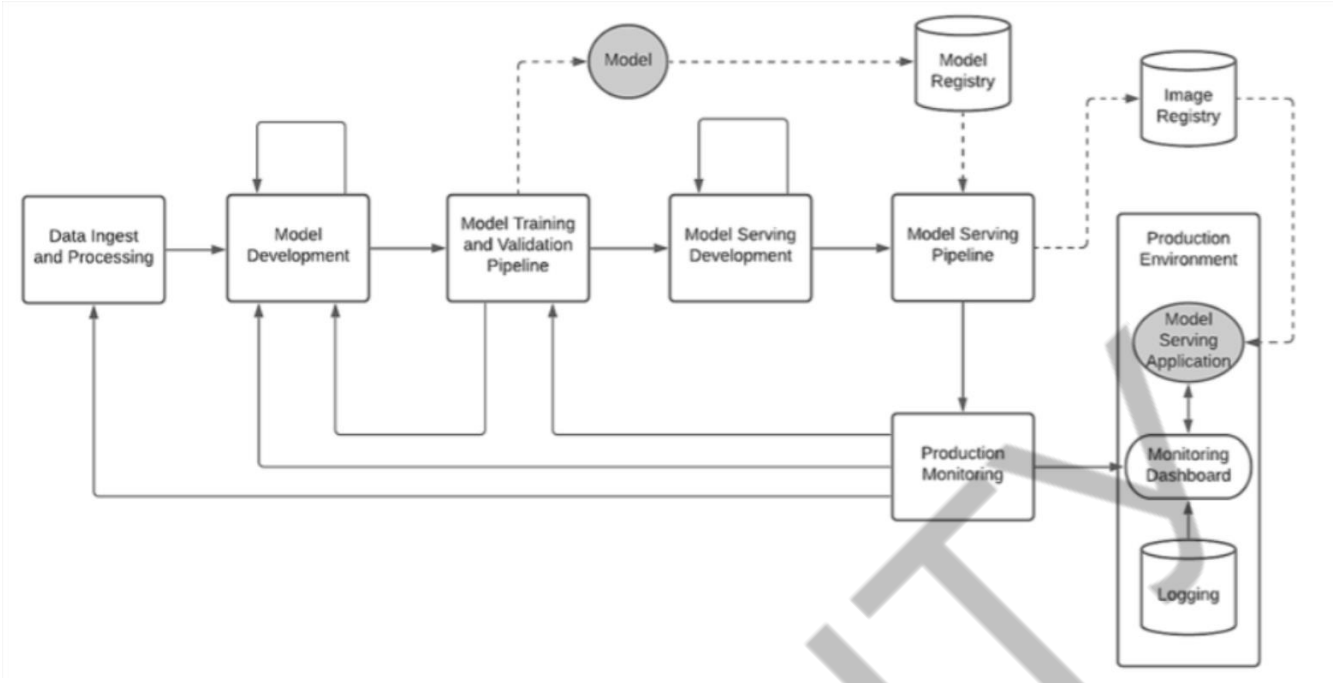

Fig. 6. Typical branching of the classification of types of testing [7]

IV. CONCLUSIONS

Using a step-by-step analysis of the available testing techniques and examining the largest machine learning models, we came to the appropriate conclusions and formed the appropriate recommendations for testing complex ML models:

1. Testing of ML models includes procedures that check whether the developed algorithm answers the request, which is formed depending on the business goals of the project. This means that the loss metrics of the ML algorithm (MSE, log-loss, etc.) must correlate with the business impact metrics (revenue, user acquisition, etc.). To measure the relationship between loss rates and impact rates can be measured through small-scale A/B testing using an intentionally degraded model.

2. It is recommended to test older models. A Machine Learning model is considered obsolete if it does not contain up-to-date data and/or does not meet business impact requirements. This is where A/B testing with outdated models can help to understand how often ML algorithms should be retrained. It is advisable to estimate the cost of more complex machine learning models by comparing the performance of, for example, a deep neural network with a simple linear regression.

3. To test the performance of the model, it is recommended to separate the commands and procedures that collect training and test data to remove dependencies and avoid propagating false methodology from the training set to the test set. This requires an additional test data set that does not overlap with the training and validation sets. This test dataset should only be used for final evaluation.

4. In order to assess the fairness of the model, it is necessary to collect as much data as possible, including potentially underrepresented categories, additionally checking the incoming features. Here, simple unit testing is suitable for creating any features, training and testing an ML model.

Also, we must define a step-by-step process for testing the ML infrastructure:

1. Training of ML models should be reproducible, that is, on the same data, it should give identical results. Various testing of machine learning models relies on deterministic learning, which is difficult to achieve due to the non-convexity of machine learning algorithms, random seed generation, or distributed learning of a machine learning model.

2. First you need to define the non-deterministic parts of the model training code base and minimize the non-determinism. Then check the stress testing using the ML API. Conduct unit tests to randomly generate input data and train the model with a single optimization step, such as gradient descent.

3. Crash tests should be performed to train a model that needs to recover from a checkpoint after a crash within training. Having checked the algorithmic correctness, you can proceed to the next step - unit testing, which is intended not to complete the training of the ML model, but to iteratively train it and reduce losses. It is recommended that you avoid testing differences with previously created models, as such tests are difficult to maintain.

4. The entire ML pipeline must pass integration testing through a fully automated test that runs it regularly. The test should confirm that the data and code successfully complete each training step and that the resulting ML model works as expected. All integration tests must be completed before the Machine Learning model gets into production. After running the model in production and before servicing it, the model should be validated by setting a threshold to check for degradation across different versions of the validation dataset. It is also necessary to check that the machine learning model is successfully loaded into the production environment, the prediction based on real data is generated properly. Ideally, the application of the Machine Learning model to the training and real samples should give the same prediction.

Despite the fact that the ML testing methodology has not yet stood firm, you can use popular tools to conduct it, such as:

– Jira for project management. In it, you can plan development, decompose tasks, and fix defects in an ML service.

– Test Rail for Quality Management. It allows you to conveniently store test scripts for the system, manage their runs and upload various testing reports.

– GitLab as a version control system.

– Jenkins for CI/CD and running autotests.

Prospects for further work are the development and creation of a complex system using machine learning technology for military rebuilding of buildings in Ukraine and the practical implementation of software testing based on selected work results [10].

REFERENCES

1. Sample Test Strategy Templates [Електронний ресурс]/ Режим доступу: https://www.softwaretestinghelp.com/writing-test-strategy-document-template/

2. Операции машинного обучения (MLOps) [Електронний ресурс]/ Режим доступу: https://azure.microsoft.com/ru-ru/products/machine-learning/mlops/

3. Software Quality: understanding the different types of software testing [Електронний ресурс]/ Режим доступу: Software Quality: the different types of software testing • Tuleap

4. Testing documentation [Електронний ресурс]/ Режим доступу: https://www.guru99.com/testing-documentation.html
5. Види тестування та відмінності між ними. Шпаргалка з т естування[Електронний ресурс]/ Режим доступу - https://qagroup.com.ua/publications/vydy-testuvannya-ta-vidminnosti-mizh-nymy/
6. Що [Електроннийресурс]/Режим доступу: https://inlnk.ru/voRYze
7. Життєвий цикл тестування ПЗ [Електронний ресурс]/ Режим доступу: Глосарій лекції №4 «Життєвий цикл тестування ПЗ» з курсу «Основи тестування ПЗ» | Безкоштовний онлайн-курс від компанії QATestLab
8. Кафедра комп'ютерних та інформаційних технологій та систем. Полтавський національний технічний університет імені Юрія Кондратюка. Основи тестування ПЗ. 2017 р. - 57 с.
9. Best test management tools of 2021 [Електронний ресурс]/ Режим доступу: https://theqalead.com/tools/best-test-management-tools/.
10. MLOps: What It Is, Why It Matters, and How to Implement It [Електронний ресурс]/ Режим доступу: https://neptune.ai/blog/mlops
11. В. Зміївський. «Перспективи застосування технологій штучного інтелекту в повоєнній відбудові україни». Proceedings of VIII International Scientific and Practical Conference Lviv, Ukraine 29-31 October 2022 – 291-297c.