

**Evaluating the quality of ML classification algorithms for 17 different classifiers from Spark ML, Keras, and Scikit-learn to detect or minimize ML bugs at an early stage before a model is deployed. This can be achieved by testing the Code, the Model, and the Data and evaluating the individual classifiers using ML quality attributes using three popular classification datasets**

- The goal is how open-source ML systems should be tested using the state of art solutions [i.e model behavioral testing](#) to build user confidence in using these systems in operational systems

## **The implementation of th project requires addressing several questions**

Our objective is to determine ways to improve ML systems quality by testing the data, model, and code using **quantitative and qualitative metrics**. These metrics are **performance, reproducibility, correctness, robustness, explainability**. **Therefore, the experimentation(implementation) must be able to answer the following questions. Hence, you need to think how to design to address each question**

- What are the most appropriate or ideal classifiers for the problem at hand, what are the most effective evaluation metrics, and what makes the classifier perform best?
  - Precision, Recall, Accuracy, ROC, confusion matrix, classification report
- Which classifiers are **robust** enough for data transformations such as data shuffling of the training instance, adding adversarial examples, and scaled data? Or Which classifier is **robust** to slight changes in the input data or for synthetic datasets? Also,
  - What are the **main factors or parameters** that contribute to sensitivity?

- What are methods or model parameters(if any) to make the **black box decision-making process** more explainable and which classifier output is **explainable** and **interpretable**? i.e
  - Explainability **in their native form** (without using any explainability tools) and using explainability tools(SHAP and LIME)
    - For example, decision making of decision tree is easy to understand at a high-level(if-else statement)
    - How do the input features contribute to the model output?
- What are the main factors/parameters/methods that enhance ML reproducibility and why is model reproducibility difficult to achieve? And
  - Which classifier is reproducible and why?
- What are the most **appropriate** classifiers, ideal **performance metrics** for the raw data (without any transformation), **unnormalized data(cleaned and transformed data but not normalized)**, **normalized data(cleaned, transformed, and normalized)**, and **imbalanced data**?
- Which combination of qualitative and quantitative metrics: performance, robustness, correctness, reproducibility, and explainability should ML practitioners consider or give priority to getting a holistic view of the model behavior before they deploy a model?
  - Why does accuracy alone not provide a complete picture of the model?
  - Is it possible to tell how each of the metrics correlates with the classifiers?
    - For example, does a decision tree classifier emphasize robustness over explainability?
- Do we get the same results using various classification models by applying the same data processing, the same or similar model parameters/hyperparameters, and other settings remaining the same?
  - We need to provide valid reasons for both yes and no answers
- What are the unique challenges of model behavioral testing when applied to classification models from Scikit-Learn, Keras, and Spark ML?
- How can we adjust the workflow to handle data and concepts drifts?

- It is possible that data and concepts, i.e the target instance may change over time and affect the quality of the ML system (model quality), for example, the prediction power. What are the best practices to minimize this effect?
- The analysis must be performed between classifiers within the same library and between libraries(focus should be here). For example,
  - Scikit-Learn linear SVM with other classifiers in Scikit-Learn and with Spark ML Linear SVM classifiers
  - We have **17 classifiers/algorithms** to be evaluated or analyzed
    - **Spark ML = 8, Scikit-Learn = 8, Keras =1**

### Tools and Technical composition

- Programming Language and IDE: **Python, Jupyter Notebook**
- Development OS: **Ubuntu ( as long we using Jupyter no problem)**
- Development Approach: **Test-Driven Development**
- Program constructs: **Classes and Functions (I love functions)**
- Required skillsets: the project is quite challenging
  - Someone who have done projects in:
  - **ML, ML Testing and quality assurance,EDA, ML Workflow orchestration(tracking), ML Model Behavioral testing ,etc**
- ML model properties to be evaluated **practically**
  - **Performance, robustness, reproducibility, correctness, explainability and Interpretability**
- ML Frameworks to used for the implementation: **Spark ML, Scikit-Learn, and Keras**
- It is very crucial to clearly understand the differences between **Model Evaluation and Model Testing** as well as **ML evaluation and ML testing**
  - The primary focus of this projecti **model testing**, not model evaluation
    - Model evaluation using performance metrics mainly depends on the performance metrics of the model, whereas **model testing is quite far beyond that** [Writing Test Cases for Machine Learning systems - Analytics Vidhya](#)

## Classification Algorithms

We have selected 16 classifiers that have the same mathematical intuition both in Spark MLlib, PySpark and Scikit-Learn. We also have one general classifier from Keras. We want to evaluate the **Keras classifier** with the rest of the classifiers. The classification algorithms are: Note each algorithm has two versions: 1 from PySpark and 1 from Scikit-Learn

- **Linear SVC or SVM.SVC(2)**
  - Max iterations, C = regularization strength, penalty(loss), fit intercept, random state
- **Logistic Regression(2)**
  - solver, penalty, C (regularization strength), max iteration, random state
- **Decision Trees(2) & Random Forest(2)**
  - Max depth, number of estimators, impurity measure, max features, bootstrap technique, random state
- **Gaussian Naive Bayes (2)**
  - Smoothing, model type(Multi, Gaussian, Bernoulli)
  - The APIs do not provide many hyperparameters as it generalizes well
- **GBTClassifier(Scikit-Learn)(1) and GBTClassifier (spark ML) (1)**
  - Max features, number of estimators, max depth, learning rate, loss/loss type, bootstrap technique, random state
- **MLPClassifier(2)**
  - hidden layer size, activation, solver, max iterations, learning rate, batch size, alpha(regularization), random state
- **One-vs-Rest(2)**
  - estimator(baseline estimator), number of parallel jobs
- **Keras Classifier(1)**
  - Binary or multi-class general classifier, random state
- **HyperParameter selection**

- The hyperparameters should be selected in such a way that they will significantly contribute to the quality of MLmodels. Considering three to four hyperparameters for each classifier seems reasonable.
- The set of hyperparameters should be present equally in Scikit-Learn and Spark ML. Otherwise, the comparison would not be reasonable

### 3 Datasets for the experimentation

**We need to use three of them to reach on conclusion**

- Titanic - Machine Learning from Disaster | Kaggle
- Pima Indians Diabetes Database | Kaggle
- Fashion MNIST | Kaggle

### Components to be Implementation

#### Data Preparation and Feature Engineering

- Apply the same or similar methods for data processing for PySpark , Scikit-Learn and Keras to evaluate the classifiers
  - Pandas as PySpark and Scikit-Learn have similar implementations
- The data preparation in Spark ML and Scikit-learn must be the same
  - We can use **Pandas Dataframes** or **Spark DataFrames** to maintain the same data processing
- Use one data integrity checking tool to find issues in the data
  - **Great Expectation, Data linter, DeepChecks,etc**
- Explanatory Data Analysis also possible

#### Data cleaning and feature engineering

- Data Cleaning tasks such as **but not limited**
  - Missing Values, Null values, Outliers, duplicates
  - Mixed data types in a single column
  - Unnecessary features for the model
- Data transformations (**not an exhaustive list**)

- Class imbalance, use
  - Oversampling and undersampling techniques
- Encoding categorical values
- Data normalization
  - To evaluate how a classifier is sensitive to normalized and unnormalized data
- Feature Importance
  - Important for testing the robustness of the model
- Perform all the required data cleaning , feature engineering

“You are free to handle this module in the manner that is most convenient for you, so long as the data is clean and appropriate for the model. The data preparation technique we use can have a big impact on the model output in many ways”

## Testing: Data, Code, and Model Testing

### Pre-Train Tests for Improving Data Quality: Data Testing

- Testing to catch bugs before we run the model
  - Identify data-related bugs(issues) early in the pipeline
- Enhancing the quality of data before feeding it to the models by writing assertions on its various features.

### Example of checks that we need to perform (not an exhaustive list)

- Check the shape of the model output and ensure it aligns with the labels in the dataset.
  - Test output range and input range of the features, for example
    - For a binary classifier, the label in the dataset must be 2
- Check for label leakage between training and validation datasets
  - Test data leak between **train** and **test** sets i.e to check duplicates
- etc

## Post-train tests to test performance, robustness, correctness, etc( Model Testing)

- Testing the trained model based on behavioral testing of ML to
- Ensure expected learned behavior is there, model performance, and implementation is correct    Testing the learning program of the trained model

### The following are the tests we need perform to the trained model

- **Minimum Functionality Tests or Adversarial testing**
  - To see how the model would perform under edge cases, bias, noise, etc.
- **Invariant Tests**
  - Modify values of less relevant features and check the prediction(↑↓)
- **Directional Expectation Tests**
  - Change the value of relevant or important features and check the prediction direction(↑↓)
- **Model evaluation to ensure satisfactory performance**
  - Often called model Inference testing
  - [Drifter ML](#) can be a better library

In **Invariant** and **Directional expectation tests** the plan should be to test the robustness of the model by changing both relevant and less-relevant features and checking how the model reacts.

## Model Pre-train and Post-train tests Clarification

- The pre-train and post-train tests vary from model to model. For example, here are useful resources for **decision trees** and **the Random forests model of pre-train and post-train tests**.
  - [How to Test Machine Learning Code and Systems](#)
- The pre-train and post-train tests are dependent on the type of data, the model, the hyperparameter, the evaluation metrics, feature values, etc. Thus, use the optimal set of tests to assure quality and reduce ML bugs that can result from Data, Code, and Model

## Code Testing

- To test the functionality of the individual functions are as expected and how the various component communicate
  - Unit testing data processing, model training, evaluation methods, etc
- Overall, the goal of the test module is to test the Code + the Learning Program (Model) + Data to improve the quality of the entire ML pipeline.
- Testing Machine Learning Systems: Code, Data and Models - Made With ML

## Training and Evaluation and track the workflow

The objective is to construct, evaluate and tune the various models. In addition you need to track the hyperparameters, visualizations, performance scores, etc using MLflow.

- **Baseline models**
  - Construct a baseline model without tuning model parameters
- **Optimized models**
  - Construct an optimized model with tuning such as Tuning Hyperparameters, Important features, K-fold Cross-validation
- **HyperParameters Tuning Method**
  - **GridSearchCV** method to find the optimal set of hyperparameters
- Compare and contrast the **optimized models** using **performance, robustness, correctness, explainability, and reproducibility**
- Summary of prediction accuracy of the **optimized models**
  - Accuracy, Classification report, Confusion matrix, ROC, Recall, Precision



## In-depth Analysis and Findings for Report Writing

The goal is to select a model that best fits the validation set using the qualitative and quantitative ML properties. **Practically we need the models using**

- **Performance Metric**

- Accuracy, classification report, confusion matrix, ROC Curve, and relevant metric.

What is the best metric for the **models** and **datasets**

- The right metric plays a crucial role in ensuring quality and reliability

- **Reproducibility**

- The objective is how to get a reproducible model or the entire ML system
- Intuitively, reproducibility is attained when we train and evaluate a model using the same configuration

Possible ways to achieve reproducibility(**not exhaustive list**)

- Setting a global seed value for the random\_state seed parameter for the train-test split and adjusting similar model parameters
- Using the pickle object to save and load trained models
- Workflow orchestration tools must be integrated to ensure reproducibility by tracking hyperparameters, performance scores, visualizations, model pickle files , etc
  - **MFlow** would be the right candidate as it easily integrates with Jupyter notebook + I have experience with that

**Feel free to use any other tool or technique for controlling reproducibility**

- **Explainability**

- The goal is to understand how model decision-making (prediction) is made and which feature contributes most to the model output.
  - LIME and SHAP would be a good candidate here
  - [Explainable AI with TensorFlow, Keras and SHAP | Jan Kirenz](#)

- **Robustness**

- The goal is to measure how robust a model is to slight changes in the data and parameters or data transformations
- How the model reacts to a slight change in the data

- Insights from the **Invariant** and **Directional Expectation tests**
  - Testing using synthetic data(modified data) could be also another approach.
  - Shuffling the training and test data and see how the model reacts
    - Any random method to make data shuffle
- **Correctness**
  - Do classification models predict the correct outcome?
  - Use any appropriate metric for measuring the correctness of the classifiers
    - I am not sure how to measure and enforce correctness

## ML Workflow Orchestration and Code Quality

- ML orchestration tools to automate and manage workflows and pipeline
  - Track hyperparameters, visuallization, performance scores
- Scripts to manage the various tasks (training, evaluation, feature engineering)
- Enforce coding standards to avoid code breakages
  - Linting tools to ensure code consistency such as **pylint**
  - For example, Jupyter Notebook extensions
    - To improve code structure, and adhere to coding standards
- Save trained model and cleaned data for later use
  - To distinguish between the raw data and the cleaned one.
- For ML workflow and individual task orchestration you can use your convenience orchestration tool that can easily integrate with Jupyter Notebook including
  - [MLflow](#): An open-source platform for the machine learning lifecycle
    - **My favorite one( I recommend you to use this one)**
  - [Apache Airflow](#)
  - [ZenML](#): Open-source pipeline framework that integrates all ML tools
  - Your favorite orchestration tools can be used, but they should be open-source and compatible with the Jupyter Notebook environment.
    - **Because I need to reproduce the experiments for my final thesis presentation(thesis defense)**

## Final Deliverables

- Implementation code and report of at least 15 pages
- The implementation code and set of instructions on how to execute or reproduce the work: **Notebook files** and python file for writing the script
- Tool for report writing: LaTeX (overleaf) and using a word document is also fine

## Report Writing Guidelines

- The description of the report must focus on the various ML properties. For example, how code, data, and model testing can improve the quality of ML systems.

### Should include the following core points

- Introduction about the **rationale behind conducting the experiments**
  - It is possible to explain the purpose of the experiment on **1** page
- **A flowchart diagram** to show the workflow and the integration of various components: tools, algorithms, metrics, data processing, etc
  - This is useful to reflect the overall workflow of each experiment we conducted and estimate the effort we invested.
- The report must entirely focus on the comparative analysis of the various classification models and the rationals behind the classifiers
  - Detailed comparative analysis using the **performance, reproducibility, explainability, robustness, and correctness properties** supported by visualizations
  - What are the **main contributions** and **impacts** on industry and academia
  - Therefore, It is expected that 90% of the reports will be here
- In my opinion, **3750 words are enough** to write a report for the experiments, **however**, the number of pages can exceed 15 pages because
  - We need to include many visualization diagrams

- The bare minimum requirement for the number of pages for the report is **15**. I am saying **the bare minimum requirement**. At least I'm expecting 15 pages where there will be a lot of visualizations in the report

## Very Important to Consider

- **At a very high level, the project seems simple as well easily attainable, however, if you carefully go through all the instructions the task is challenging and demanding. The project is doable but could take a lot of effort and time.**
- Hence a Machine Learning Engineer with the following set of skills would be a perfect fit for the task
  - ML LifeCycle, Model **behavioral testing**, **model** pre-train tests, model post-train tests, **Minimum Functionality Tests**, **Invariant tests**, **Directional expectation tests**, ML Software testing, Explanatory Data Analysis, ML Workflow Orchestration, and a solid practical experience with how to measure **performance, robustness, explainability, correctness, and reproducibility** of classification algorithms
  - How to use effectively **classification models from Spark ML, Scikit-Learn, and Keras** and how these algorithms work internally.
- The in-depth analysis for the report must focus on **performance, robustness, reproducibility, explainability, and correctness**
  - i.e which model/system is robust, reproducible, and explainable? And the rationales behind the outcomes or
  - How reproducibility , explainability can be assured in ML systems?
- I often used the phrase **"not exhaustive list"** in many of my writings.
  - I wanted to give an example or a direction on how to tackle that specific task. However, there are many tests or checks that should be taken into consideration besides those mentioned.
  - **You need to consider many possible scenario**

- I am always online to clarify anything in the development process. When there is some misunderstanding. Do the following things accordingly
  - I encourage you to reach out to me at any time
  - Use the optimal approach in ML development and testing ML components
- I included some tools such as **Drifter-ML** and a few terms about testing techniques such as model behavioral testing, **Invariant testing**, and **model pre-train tests** for two main reasons
  - To give you a bit of information or direction
  - The whole thesis depends on these concepts, tools, methods, etc
- **Again ML evaluation is different from ML testing**
  - **The focus of the work on testing where performance is one metrics**
- The instruction must be followed AND READ THEM CAREFULLY
- **COPYING CODE FROM THE INTERNET IS TOTALLY PROHIBITED !!!**
  - This is because, I gave the task to one company before, but what they did was just copy the code from the internet and they send it to me as draft.
- We have **17 classifier**, **3 datasets** from **3 machine learning libraries**, and many tests.
  - We need to write many test cases
- **Refer to the links below whenever you get confused**
  - I can fairly say that, many of the tasks are answered in the resources
- **I need a draft in 2 days**
- **Due date: 27 at 12:00 CET(Central European Time)**

## Well Curated resources for Implementation

[How to Test Machine Learning Code and Systems](#)

Model behavioral testing, model pre-train tests, and model post-train tests of classification models using the famous Titanic datasets

[Checklist — Behavioral Testing of NLP Models | by Khuyen Tran | Towards Data Science](#)

Minimum Functionality Tests, Invariant Tests, Directional Expectation Tests

[How to Trust Your Deep Learning Code | Don't Repeat Yourself](#)

An example of testing every piece of ML systems code to build user trust

[MLOps with ZenML and MLFlow: how can we build a model training pipeline? — a practical example | by Kattson Bastos | Aug 2022 | Medium](#)

[GitHub - eugeneyan/testing-ml: 🔍 Minimal examples of machine learning tests for implementation, behavior, and performance.](#)

[Testing Machine Learning Systems: Code, Data, and Models - Made With ML](#)

Testing ML Systems by testing the Code, Data, and Models

[Effective testing for machine learning systems](#)

Model testing and model evaluation, model pre-train and post-train tests

[Open the Black Box: an Introduction to Model Interpretability with LIME and SHAP - Kevin Lemagnen](#)

[Machine Learning Testing for Beginners - All in One Guide - TestProject](#)

Minimum Functionality Test, Directional Expectation test, Invariant test

[Testing your machine learning \(ML\) pipelines | Into the depths of data engineering](#)

[Testing ML Code: How Scikit-learn Does It | by Moussa Taifi Ph.D. | Analytics Vidhya | Medium](#)

[How to get absolutely reproducible results with Scikit Learn? - Stack Overflow](#)

[Jose Quesada - A full Machine learning pipeline in Scikit-learn vs in scala-Spark: pros and cons](#)

[ML Model Interpretation Tools: What, Why, and How to Interpret - neptune.ai](#)

[Reproducible ML: Maybe you shouldn't be using Sklearn's train\\_test\\_split | Engineering for Data Science](#)

[Article Review: The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction by Google](#)

What needs to be tested for ML systems(Code, Model, data, and features)

<https://serokell.io/blog/machine-learning-testing>

[Behavioral Testing of NLP models with CheckList](#)

An excellent introduction to model behavioral testing

[Writing Test Cases for Machine Learning systems - Analytics Vidhya](#)

Model pre-train and post-train tests for scikit-learn using Pandas.

[https://github.com/Khushee-Upadhyay/Testing\\_Demo\\_Project](https://github.com/Khushee-Upadhyay/Testing_Demo_Project)

Pre-train and Post-train tests for an insurance prediction problem

[Testing Machine Learning Systems: Code, Data, and Models - Made With ML](#)

An Excellent resource for Testing ML Code, Data, and Model

[PyTest for Machine Learning — a simple example-based tutorial | by Tirthajyoti Sarkar | Towards Data Science](#)

How to use Pytest from scikit-learn to test features(data) and models

[How To Unit Test Machine Learning Code - KDnuggets](#)

[How to Test Machine Learning Models | Deepchecks](#)

Model Evaluation vs. Model Testing differences, Robustness, Interpretability, Reproducibility, Correctness, Testing Trained Models, Invariance Tests, Directional Expectation Tests, Minimum Functionality Tests

[Beyond Accuracy: Behavioral Testing of NLP Models with CheckList - ACL](#)

[Anthology](#) Foundation of Model Behavioral testing