What is JanIA?

JanIA Intelligent Decisions:

- It is an Artificial Intelligence (AI) solution applied in the field of quality assurance, control and quality engineering of information systems.

- It is based on the exploitation of all the information generated by the software processes and environments for the intelligent governance of the applications.
Which benefits does JanIA provide?

- Reducing the number of incidents in production
- Reduction of maintenance costs.
- Reduced time-to-market of applications.
- Avoiding "unnecessary" tests
Which capabilities could JanIA offer?

**Testing Optimization**
- Functional testing
- Non-functional testing
- Regression testing
- Acceptance testing

- Recommend testing strategy
- Predict test results
- Prioritize regression tests

**Software Risks View**
- SW bugs
- Improvement points

- Predict the risk level of a release
- Predict failure-prone SW components
- Correlate code quality with production issues
JanIA, how is it structured?

Static SW Metrics
- KLOC
- V(G)

SW Promotions
- Dates
- Incidents

Pre-Production testing
- Test cases defined
- Test cases executed
- Defects encountered

Historic data from +9 months

Machine Learning
- Spatial Clustering (DBSCAN)
- Clustering KMEANS
- Spectral Clustering
- Hierarchical Clustering
- Gaussian Classifier (Gaussian Naive Bayes - GNB)
- Support-Vector Machine (SVM)
- Decision Tree
- Linear Discriminant Analysis (LDA)
- Support Vector Regression (SVR)
- Linear Regression

Deep Learning
- Convolutional Neural Network (CNN)
- Long Short-Term Memory Neural Network (LSTM)

CI/CD
- Quality Gates (medium term)
- User friendly DashBoard (deliverable)
Feasibility analysis to apply JanIA

Feasibility Analysis: Proof of Concept

Objective:

Explore the feasibility of applying Predictive Modeling Techniques (based on Machine Learning) for Risk Assessment associated with different software applications in different aspects such as Quality Assurance or Control.

• Data availability? Representativeness, Volume, Variability, Consistency

• Predictive capacity? Target variables/ KPIs, Mathematical Models

• Favorable results? Confidence, Alignment, Improvement capacity
REAL USE CASE (POC)

- Context and motivations

- Use Case 1: Accessibility (ACCE) and Vulnerability (VULNE) Test Prediction

- Use Case 2: Performance Test Prediction
Real use case (POC)

Context and motivations:
- Project environments

Development Environment:
- Large number of versions to promote monthly.
- Manual and expensive pre-exploitation tests to implement.
- Small testing team.

QA/QC Environment:
- Bottleneck between Pre-exploitation and Production environments.
- High time-to-market.
- Applications in production with errors.

Production Environment:

Development
Pre-exploitation
Production
Real use case (POC)

Use Case 1: ACCE and VULNE Test Predictions

Objective: Predict the result of the Accessibility and Vulnerability Tests of a version of an application taking as input a set of data composed of information on the version itself and its previous versions.

Initial data
- Collecting data on the versions and applications to be analyzed

Expected result
- Test results
- Confidence in prediction
- Prioritization of the tests to be executed
Use Case 1: ACCE and VULNE Test Predictions

- **Problem Formulation**
  - **Input Features $X$:** The feature vector $x^{(i)}$ is formed by 31 variables obtained from the data provided by a static code analysis tool together with data from previous versions of the applications themselves.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>91</td>
<td>...</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>100</td>
<td>...</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Training set size:**
  - Number of versions promoted to Pre-Exploitation environment (~2500)

- A version promoted to pre-exploitation corresponds to an example in the training set.
Real use case (POC)

Use Case 1: ACCE and VULNE Test Predictions

- Problem formulation
  - Output Labels $Y$:
    - Test Result ACCE and VULNE: These are discretized into binary classes, Test Accepted and Test Rejected which correspond respectively to classes 0 and 1 used in ML models.
    - Recommendation probability: Value between 0 and 1 that represents the exact probability offered by the model to assign one class or another.
Use Case 1: ACCE and VULNE Test Predictions

- Training dataset generation
  - For the generation of the train dataset, application data has been taken from 2016 to 2022. Among them are:
    - Automatic labeling and preprocessing.
    - Health factors
    - Defects by characteristic and priority
    - Code Metrics
    - Other explanatory variables

Information about version labeling and preprocessing:
As is well known in supervised learning models, input data must be labeled. In this case, this process is automatic, which is an advantage when applying the model, since it is not necessary to invest time on it.
Use Case 1: ACCE and VULNE Test Predictions

Model training:
- Items to have in mind
  - Binary classification problem
  - Unbalanced dataset
  - Conservative vs Bold Strategy
  - Feature importance

ML Models checked out
- Algorithm selection:
  - Logistic Regression
  - XGBoost
  - Random Forest
- Data Split (rebalanced):
  - Training on 85%
  - Validation on 15%
Real use case (POC)

Use Case 1: ACCE and VULNE Test Predictions

### Results

- **ACCESSIBILITY TESTS**
  - Not prioritizing tests in 21.2% of the versions. Accessibility problems detected with a 91% accuracy.
  - Prioritize tests in 52% of the versions. Accessibility problems detected with 86% accuracy.
  - Moderate/Low confidence 26% of versions

- **VULNERABILITY TESTS**
  - Not prioritizing tests in 28.7% of versions. Vulnerability problems detected with a 97% accuracy.
  - Prioritize testing in 27.2% of versions. Vulnerability problems detected with 82% accuracy.
  - Moderate/Low confidence 44.1% of versions
## Real use case (POC)

### Use Case 1: ACCE and VULNE Test Predictions

#### Results

<table>
<thead>
<tr>
<th>Apps</th>
<th>Version</th>
<th>Vulnerability</th>
<th>Recommendation</th>
<th>Decision</th>
<th>Recomendation</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRPC</td>
<td>02</td>
<td>NO</td>
<td>PDTE</td>
<td>NO</td>
<td>PDTE</td>
<td></td>
</tr>
<tr>
<td>CAPP</td>
<td>04.04.01.01</td>
<td>NO</td>
<td>PDTE</td>
<td>YES</td>
<td>PDTE</td>
<td></td>
</tr>
<tr>
<td>CVIM</td>
<td>04.04</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>Low Priority</td>
</tr>
<tr>
<td>PRPI</td>
<td>04.00.00.45</td>
<td>NO</td>
<td>PDTE</td>
<td>NO</td>
<td>PDTE</td>
<td></td>
</tr>
<tr>
<td>CIMP</td>
<td>01</td>
<td>NO</td>
<td>PDTE</td>
<td>NO</td>
<td>PDTE</td>
<td></td>
</tr>
<tr>
<td>CINI</td>
<td>01.10.00.01</td>
<td>NO</td>
<td>PDTE</td>
<td>NO</td>
<td>PDTE</td>
<td></td>
</tr>
<tr>
<td>ADGC</td>
<td>02.04</td>
<td>NO</td>
<td>PDTE</td>
<td>NO</td>
<td>PDTE</td>
<td></td>
</tr>
<tr>
<td>AGFE</td>
<td>03.01.21.04</td>
<td>NO</td>
<td>PDTE</td>
<td>YES</td>
<td>PDTE</td>
<td>Medium Priority</td>
</tr>
<tr>
<td>AIFI</td>
<td>04.25</td>
<td>NO</td>
<td>PDTE</td>
<td>YES</td>
<td>PDTE</td>
<td></td>
</tr>
<tr>
<td>AIFO</td>
<td>06.02.00.03</td>
<td>NO</td>
<td>PDTE</td>
<td>YES</td>
<td>PDTE</td>
<td></td>
</tr>
</tbody>
</table>

- **Low Priority**
- **Medium Priority**
- **High Priority**
- **Against the recommendation**
Real use case (POC)

Use Case 2: Performance Test Prediction

Objective: Predict the production behavior of a version taking as input a set of data composed of information on the version itself and the performance of its previous versions together with the defects detected in production.

Initial data
- Collecting data on the versions and applications to be analyzed

Expected result
- Production Performance
- Confidence in prediction
- Correlation with pre Performance tests
Use Case 2: Performance Test Prediction

Problem Formulation

- **Input Features** $X$: The feature vector $x^{(i)}$ is formed by 33 variables obtained from the production data of previous versions related to execution errors and various metrics such as response time or CPU consumption.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>99</td>
<td>91</td>
<td>...</td>
<td>2,4</td>
<td>350</td>
</tr>
<tr>
<td>80</td>
<td>100</td>
<td>...</td>
<td>7,8</td>
<td>129</td>
</tr>
</tbody>
</table>

A production version corresponds to an example of the training set.

Training set size:
Number of versions promoted to Exploitation environment (~3000)
Use Case 2: Performance Test Prediction

Problem Formulation

Output Labels $Y$:

- Performance in production: This value is discretized into binary classes, Good Performance and Bad Performance that correspond respectively to classes 0 and 1 used in the ML models.

- Recommendation probability: Value between 0 and 1 that represents the exact probability offered by the model to assign one class or another.
Real use case (POC)

Use Case 2: Performance Test Prediction

- Training dataset generation

  - For the generation of the train dataset, application data has been taken from 2016 to 2022. Among them are:

    - Information about version labeling and preprocessing:
      For the labeling of the versions, thresholds calculated monthly from the metrics and errors of applications in production during the previous month have been taken into account.

- Automatic labeling and preprocessing.
Real use case (POC)

Use Case 2: Performance Test Prediction

- Results

- 52.7% of the versions in the test data receive a "GOOD" PERFORMANCE prediction with a 98.5% accuracy.

- Regarding the prediction of versions with "BAD" PERFORMANCE, a 96.1% success rate is achieved and a range of 7% of the versions analyzed in the test data.

- Moderate/Low confidence 40% of versions
Conclusions

The results obtained during the POC are good enough to put the solution into practice.

The solution is currently in process of being integrated into the workflow between environments while continuing to be improved.

The solution has been extended to other areas of the company where a new proof of concept is being developed.
Any further questions?

Contact me:
pablo.manso@mtp.es