Outlr.

A web application for analyzing subspace outliers on high dimensional datasets

Bennet Alexander Hörmann, Salomo Hummel,
Simeon Hendrik Schrape, Erik Bowen Wu, Udo Ian Zucker

02.12.2022

Contents

1 Introduction

With the recent rise in popularity of machine learning, research in outlier detection becomes more and more important since outliers degrade the performance of other machine learning methods. While the most effort is being put into improving general state-of-the-art methods, there are not many platforms allowing researchers to easily investigate subspace outliers and the detection of these on high-dimensional data.
In *Outlr.* we like to introduce a modern web service, which lets users run state-of-the-art Outlier detection methods (ODMs) on specified sub-spaces of high-dimensional data sets and compare the results of the run while providing an accessible and intuitive user interface. Usually, this would require running many experiments manually on multiple subspaces, which is hard to track. With *Outlr.* we aim to improve this workflow.
The results are presented in an organized overview, where the user can decide to further process them. Every user also has access to their previously run experiments, to avoid running the same experiment more than once.

2 Product Functions

Mandatory product functions

User Management **FM1**
Implemented by: RM1 RM2 RM3 RO1 RO2 RO3 RO4 RO5 RO6 RO28
Mockups: M1 M2
The application provides features to manage users. Users create accounts to store and access previously run experiments.

Dashboard **FM2**
Implemented by: RM4 RM5 RO7 RO8 RO9 RO12 RO26
Mockups: M4
The dashboard gives users an intuitive overview of all the previous experiments.

Create experiment **FM3**
Implemented by: RM6 RM7 RM8 RM9 RM10 RM11 RM12 RO10 RO11 RO12 RO13 RO14 RO15 RO16 RO17 RO18 RO19 RO20
Mockups: M3
Users can create a new experiment by entering the required data.

Run experiment **FM4**
Implemented by: RM13 RO21 RO22 RO23
Mockups: M3
After the user created an experiment and selected all of the hyperparameters, the experiment can be run that is to execute the ODM on the given dataset.

Experiment results **FM5**
Implemented by: RM14 RM15 RO9 RO12 RO24 RO25
Mockups: M5
After an experiment has been run the application provides detailed information about its result.

Optional product functions

Landing page **FO1**

Implemented by: RM1 RM2 RM3

The first page that users see is the landing page. On this page users can register or log in and when signed in log out. The page also informs potential users about the application.

About us page **FO2**

The about us page gives users information about the developer team.

Compare experiments **FO3**

Implemented by: RO26 RO27

This page shows diagrams that compare a selected set of experiments.

3 Requirements: Scope and Delimitation

Mandatory requirements

User Management

Create account (functional) **RM1**
Tested by: T1 *Implements:* FM1 FO1
Mockups: M1 M2 M4
Users can create an account by providing a valid username and password.

Login (functional) **RM2**
Tested by: T1 T2 *Implements:* FM1 FO1
Mockups: M2 M4
Users can log into their account by entering their usernames and their respective passwords.

Logout (functional) **RM3**
Tested by: T3 *Implements:* FM1 FO1
Mockups: M1 M2 M4
Users can log out of their account.

Dashboard

View dashboard (functional) **RM4**
Tested by: T5 *Implements:* FM2
Mockups: M1
A dashboard provides an overview of all experiments that were run.

Click on experiment on dashboard (functional) **RM5**
Tested by: T5 *Implements:* FM2
The dashboard entries are clickable and lead to the corresponding experiment results.

Create experiment

Name experiments (functional) **RM6**
Tested by: T10 *Implements:* FM3
Users can name their experiments.

Upload dataset as CSV file (functional) **RM7**
Tested by: T6 T10 *Implements:* FM3
Mockups: M3
Users can upload a dataset as CSV file.

Upload ground-truth file as CSV file (functional) **RM8**
Tested by: T6 T10 *Implements:* FM3
Mockups: M3
Users can upload a ground-truth file when creating an experiment.

Select subspaces from dataset (functional) **RM9**
Tested by: T7 T10 *Implements:* FM3
Mockups: M3
Users can select subspaces from the dataset to be processed by the subspace logic.

Specify subspace logic (functional) **RM10**
Tested by: T7 T10 *Implements:* FM3
Users can specify a subspace logic using *logical or* and *logical and* operators.

Customize hyperparameters (functional) **RM11**
Tested by: T9 T10 *Implements:* FM3
Mockups: M3
Based on the selected ODM users can customize all its hyperparameters.

Provide a significant number of ODMs (functional) **RM12**
Tested by: T8 T10 *Implements:* FM3
Mockups: M3
Users can select an ODM from a significant subset of the ODMs provided by the PyOD library.

Run experiment

Run experiment (functional) *Implements:* FM4 **RM13**
Mockups: M3
After creating an experiment users can run it.

Experiment results

Display experiment results (functional) **RM14**
Tested by: T12 *Implements:* FM5
Mockups: M5
The app provides an overview page for every executed experiment which includes the following information:

- Experiment name
- ODM used for experiment
- Accuracy
- Execution date and time
- Time taken to run the experiment
- Number of detected outliers
- The indices of the detected outliers

Download detected outliers (functional) **RM15**
Tested by: T4 *Implements:* FM5
Mockups: M5
Users can download the indices of the detected outliers from an experiment as a CSV file.

Others

English language (non-functional) **RM16**
The language of the application is english.

User password is stored securely (non-functional) **RM17**
User passwords are stored using hashing and other methods for security.

Unexpected errors (non-functional) **RM18**
If an unexpected error occurs, the application will not crash and instead notify the user.

Good reliability and maturity (non-functional) **RM19**
The application is reliable and mature, which is achieved by extensive testing throughout the project.

Easy to learn and efficient workflows (non-functional) **RM20**
For the target group (see 4), the application is easy to learn and provides efficient workflows.

Good resource handling (non-functional) **RM21**
The application releases all resources of an experiment after it is run. Only the results that the user needs will be saved.

Good maintainability (non-functional) **RM22**
The source code of the application is easy to read and modify. This requires a good code structure and good documentation achieved by helpful comments.

Modern and appealing GUI (non-functional) **RM23**
The GUI of the application is designed to look modern and appealing. By default, the website has a dark theme.

Optional requirements

User management

Delete account (functional) **RO1**
Implements: FM1
Users can delete their account.

Create user with email (functional) **RO2**
Implements: FM1
Users provide an email when creating an account. Afterward, they have to confirm their email by opening a verification link sent to their email.

Login with email instead of username (functional) **RO3**
Implements: FM1
Users can also log into their account by entering their email and the respective password.

Share experiment result link (functional) **RO4**
Implements: FM1
Users can create and share a link to an experiment so that other users can see its results when opening the link.

Users can try out functions (functional) **RO5**
Implements: FM1
Users can try out the application without an account.

Forgot password feature (functional) **RO6**
Implements: FM1
When users forget their password they can set a new password after sufficient proof of identity.

Dashboard

Filter and search experiments on dashboard (functional) **RO7**
Implements: FM2
Users can filter and search the experiments shown on the dashboard.

Sort experiments on dashboard (functional) **RO8**
Implements: FM2
Users can sort the experiments shown on the dashboard.

Loading animations (functional) **RO9**
Implements: FM2 FM5
The application shows a loading animation when it waits for the results of an experiment.

Create experiment

Upload CSV file containing generated data (functional) **RO10**
Implements: FM3
Users can upload a CSV file containing generated data, which is merged with the uploaded dataset before running the experiment.

Create new experiment based on a previously run experiment (functional) **RO11**
Implements: FM3
A new experiment can be created by copying a previously run experiment. Users only need to re-upload the dataset and ground-truth file.

Name dataset (functional) **RO12**
Implements: FM2 FM3 FM5
Users can specify a name for an uploaded dataset, which is displayed in experiment results and the dashboard.

Subspace logic can be nested (functional) **RO13**
Implements: FM3
Subspace logic may contain nestings of logical operators.

Find outliers in given number of subspaces (functional) **RO14**
Implements: FM3
Users can specify a list of subspaces and a number k, so that the experiment identifies all data points, that are outliers in at least k of the given subspaces.

Use column headers (functional) **RO15**
Implements: FM3
A subspace can be specified by naming the column headers of the columns that are included in the subspace.

Users can apply their own ODM (functional) **RO16**
Implements: FM3
Instead of selecting an ODM from the list users can upload their own ODM.

Provide all ODMs from PyOD (functional) **RO17**
Implements: FM3
The app provides all ODMs from PyOD.

No ground-truth file needed (functional) **RO18**
Implements: FM3
The user does not have to add a ground-truth file to the dataset.

Drag and drop support (functional) **RO19**
Implements: FM3
The user can drag and drop the CSV files for the dataset and ground-truth file.

Dataset preprocessing (functional) **RO20**
Implements: FM3
Users have options for preprocessing on the uploaded dataset.

Provide new ODMs (functional) **RO21**
Implements: FM4
The application provides ODMs developed by our team.

Run experiment

Run experiments concurrently (non-functional) **RO22**
Implements: FM4
The application runs experiments concurrently.

Run experiments with hyperparameter range (functional) **RO23**
Implements: FM4
Users can specify a range for a hyperparameter and a step size. When running the experiment, the outlier detection is done for every step in the specified range. The application shows a diagram where the accuracy is plotted against the hyperparameter value.

Experiment results

Show ROC curves (functional) **RO24**
Implements: FM5
The application shows ROC curves for an experiment.

Display output log (functional) **RO25**
Implements: FM5
The application displays output logs of the execution at the user's request.

Select experiments to compare (functional) **RO26**
Implements: FM2 FO3
On the dashboard, users can select experiments to compare them.

Compare experiments (functional) **RO27**
Implements: FO3
The application shows diagrams that compare metrics from different experiments selected by the user. For example, the ROC curves can be displayed in a single graph or a histogram can be displayed to compare accuracies.

Others

Light theme (non-functional) **RO28**
Implements: FM1
Users can change the GUI to a light theme.

Delimitation

No mobile version **D1**
A mobile application is not planned.

Application not optimized for mobile devices **D2**
Optimizing the website for mobile devices is not planned.

No guarantee regarding the provided ODMs from PyOD **D3**
The application uses PyOD in the backend and thus the performance, correctness, and reliability of the provided ODMs are only as good as PyOD implements them.

Usability for users unfamiliar with outlier detection **D4**
For users unfamiliar with outlier detection, the application may be confusing. However, we do not aim to improve this experience.

4 Product Use and Environment

Target group

The target group are people interested in researching subspace outliers since the application provides useful tools for analyzing outliers in different subspaces and comparing different ODMs on these subspaces. The application makes it easy to run and organize many experiments using different ODMs and combinations of subspaces.

Operating conditions and environment

The client-side application runs on any PC running Windows ($\geq$ 10) or common Linux distributions. It also requires Firefox ($\geq$ 107.0) or Chrome ($\geq$ 107.0) and an internet connection.
The server-side application can be deployed to any common Linux server that is able to run a Python interpreter.

5 Product Overview

General application structure The following use-case diagram[1] shows the general application structure from the perspective of a user.

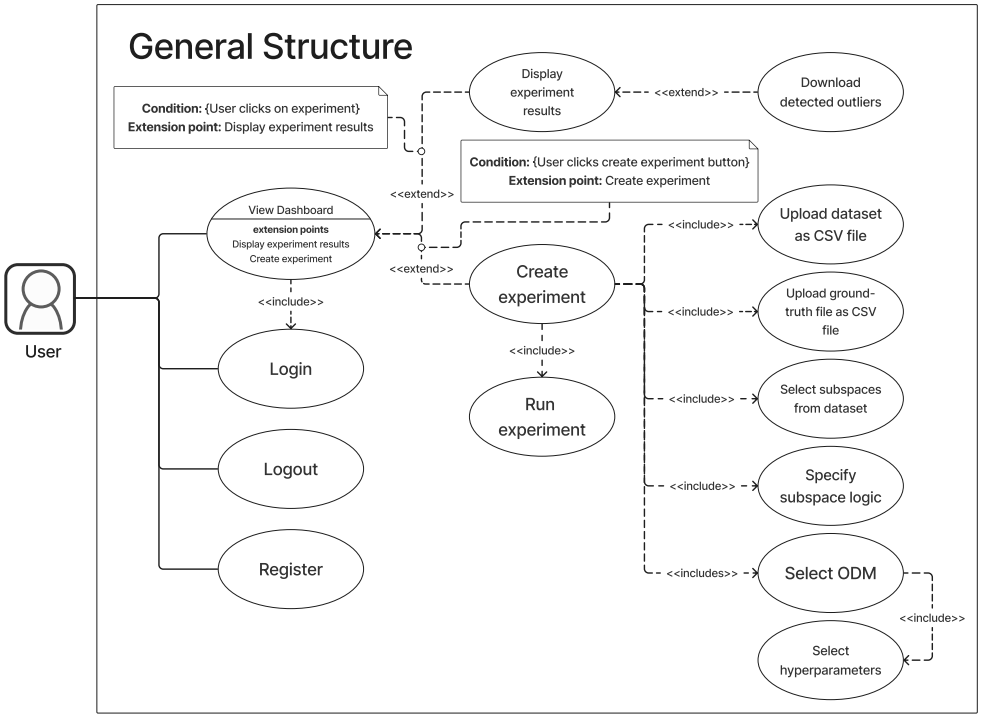

Figure 1: General structure use-case diagram

Create and run experiment The following activity diagram shows the process of creating and running experiments.

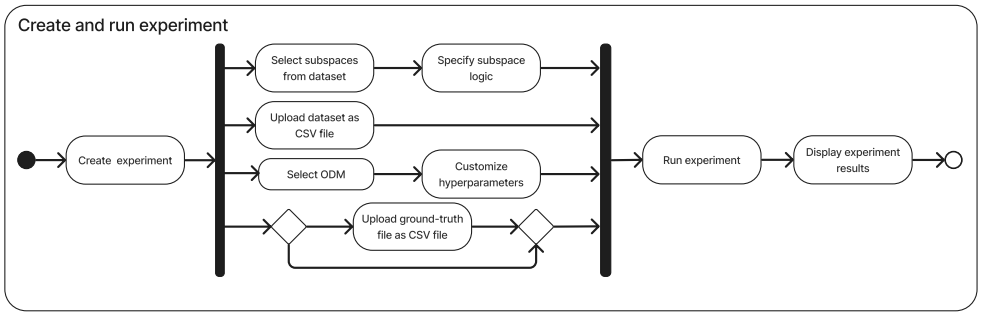

Figure 2: Experiment activity diagram

[1] To simplify the diagram, connections between the user and use-cases are omitted if a use-case extends another use-case that is already connected to the user.

6 Product Data

User profile data

- Username
- Authentication data i.e. password hash
- List of all experiments

Experiment data

- Name given by the user
- ODM used
- Hyperparameters
- Execution date
- Time taken for execution
- Number of outliers detected
- Subspaces and subspace logic
- Indices of detected outliers
- Accuracy

7 System Architecture

The application is modeled as a 3-Tier architecture consisting of a data tier, application tier, and presentation tier. The data tier and application tier run on the server. The presentation tier runs in the browser. The data tier is a SQL database storing all persistent data.[2]. The presentation tier is a web-based UI. The data tier and presentation tier do not directly communicate with each other. All communication is handled by the application tier by accessing the database and providing an API for the presentation tier. It also handles all business logic e.g. execute the ODMs, apply the specified subspace logic, etc.

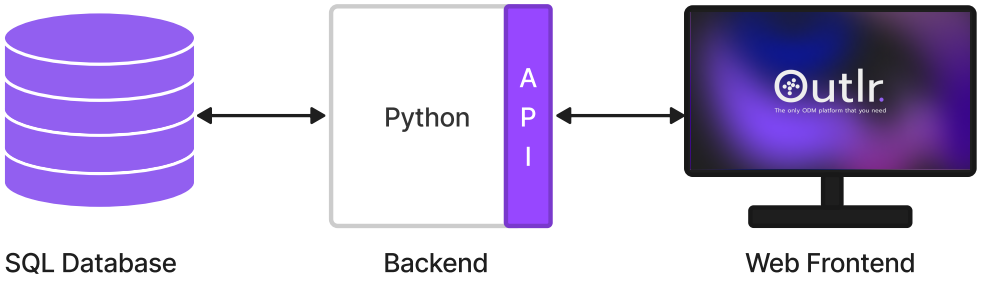

Figure 3: Basic overview of the system architecture: the SQL database represents the data tier, the backend represents the application tier and the web frontend represents the presentation tier

[2]See section 6 on product data for detailed information

8 Product Performance

The application can maintain 20 000 user accounts. The application can handle 200 users at the same time while satisfying the following minimum performance requirements:

Button response time	200ms
Page navigation	200ms
Run experiment	not defined (see D3)
Create User	5sec
Log in	2sec
Log out	2sec

Table 1: Minimum performance requirements

9 GUI mockups

The following mockups give an idea of how the GUI of the application may look like. The actual GUI may deviate from these mockups. They also contain some elements, that are part of optional requirements and are not guaranteed to be present in the application. The intention of this section is solely to highlight certain product functions and corresponding requirements.

Landing page **M1**

Product functions: FM1 FO1

Requirements: RM1 RM3 RM4

The landing page is the first page that users see.

Figure 4: Landing page when the user has not logged in yet. Elements depicted: sign up button, navigation bar, login button, try out button (optional)

Figure 5: Landing page when the user has logged in. Elements depicted: navigate to dashboard button, navigation bar, profile menu

User Management **M2**

Product functions: FM1

Requirements: RM1 RM2 RM3 RO2

Users can register and log in using the following forms.

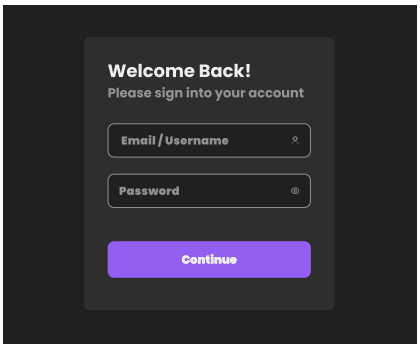

Figure 6: Login form (unselected)

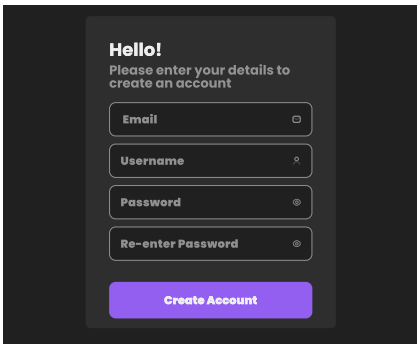

Figure 7: Register form (unselected)

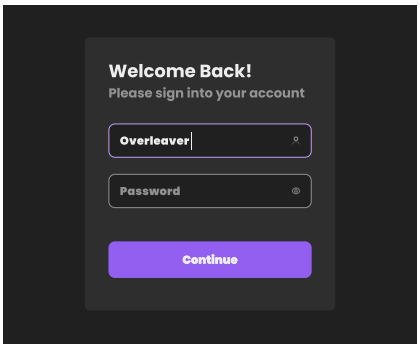

Figure 8: Login form

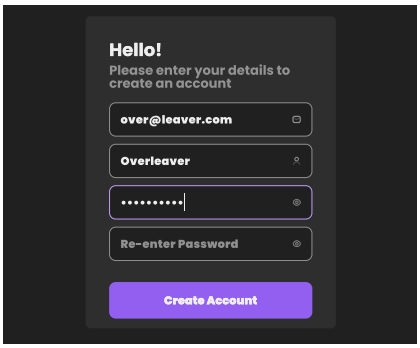

Figure 9: Register form

Create experiment **M3**

Product functions: FM3 FM4

Requirements: RM7 RM8 RM9 RM11 RM12 RM13

Dedicated page allowing users to create a new experiment.

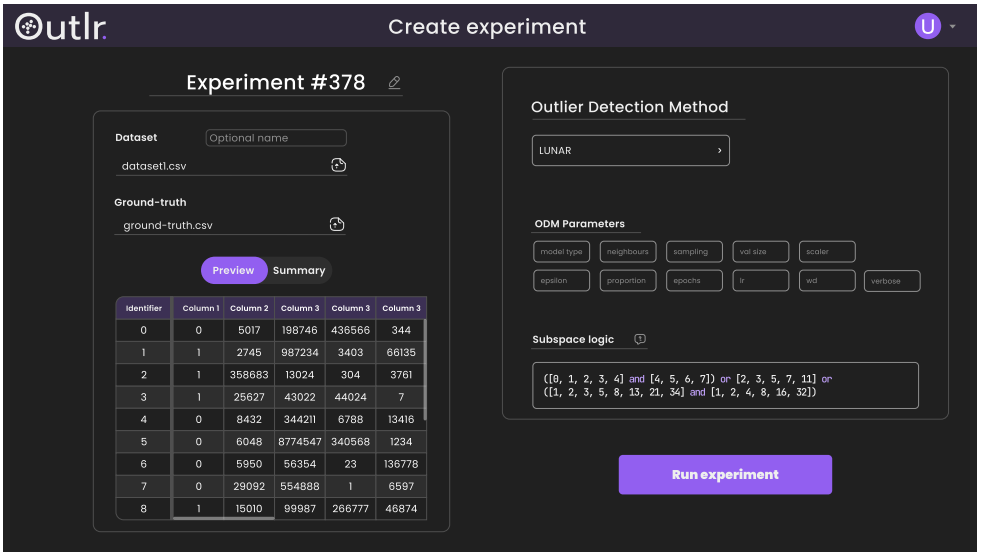

Figure 10: Create experiment page. Elements depicted: upload dataset, upload ground-truth file, dataset preview and summary, select ODM, specify hyperparameters, subspace logic, experiment name (optional), dataset name (optional), run experiment button

Dashboard **M4**

Product functions: FM2

Requirements: RM1 RM2 RM3

The dashboard gives users an overview of the previous experiments. According to optional requirements, the user may choose to refresh, compare, search, delete experiments, clear, sort, and filter the dashboard.

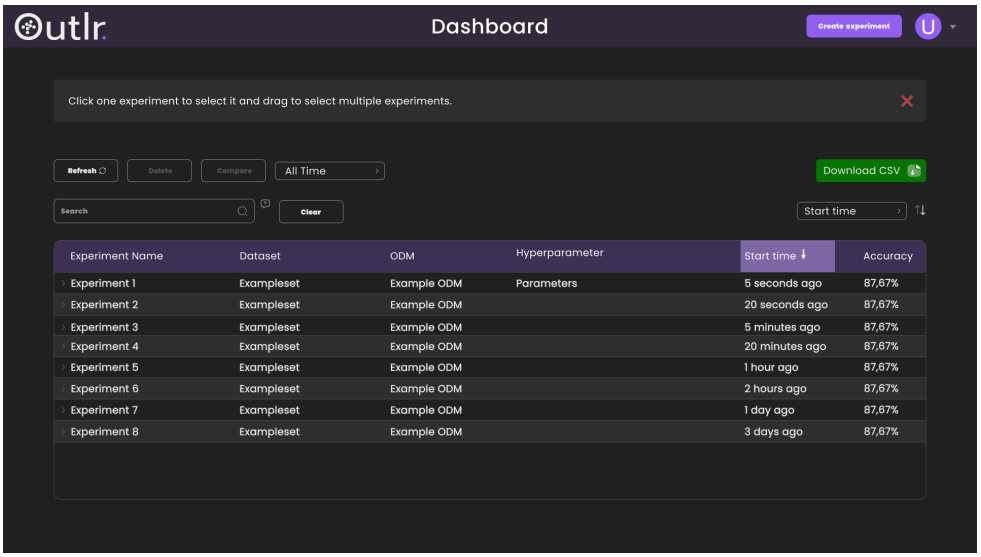

Figure 11: Dashboard. Elements depicted: table depicting previous experiments, hint box, download CSV button, refresh, delete, compare, search and clear, time span filter, sorting type button and create experiment button

Experiment result **M5**
Product functions: FM5
Requirements: RM14 RM15 RO24
Dedicated page to display experiment results.

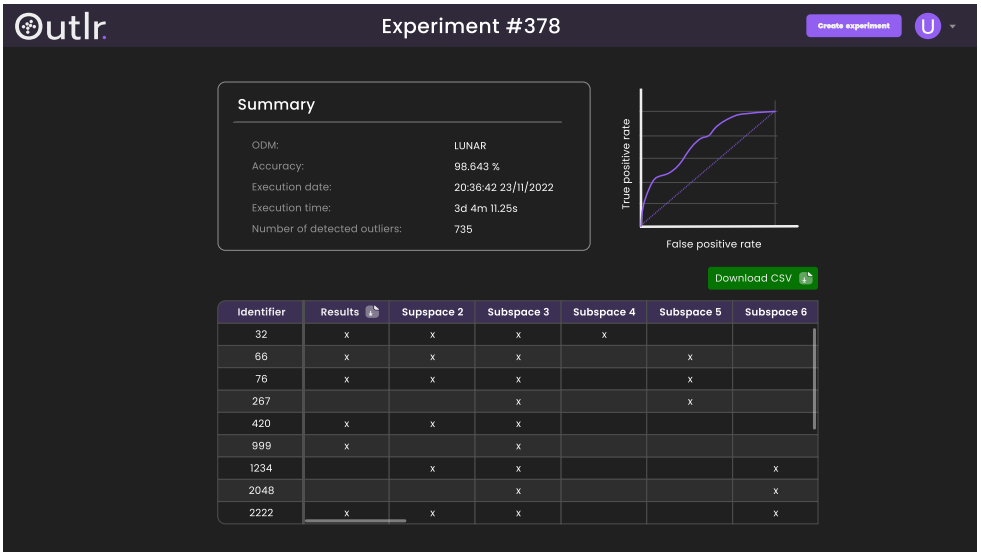

Figure 12: Experiment results. Elements depicted: experiment summary, selected hyperparameters for ODM, detected outliers and option to download them, roc curves (optional)

10 Testcases

User Management

Create new account **T1**

Tests: RM1 RM2

State: On signup window

T1.1 **Action:** The user enters valid input and clicks on the signup button.
Reaction: Success message and entry in database

T1.2 **Action:** The user enters a username that already exists and clicks on the signup button.
Reaction: Error message and prompt to reenter credentials

Login **T2**

Tests: RM2

State: On login window

T2.1 **Action:** The user enters valid input and clicks on the login button.
Reaction: Redirected to Dashboard

T2.2 **Action:** The user enters an incorrect password and clicks the login button.
Reaction: Error message, that password was entered incorrectly

Logout **T3**

Tests: RM3

T3.1 **Action:** The user clicks the logout button.
Reaction: The user gets logged out.

Download outliers as CSV file **T4**

Tests: RM15

T4.1 **State:** On experiment results page
Action: The user clicks a button to download outliers.
Reaction: A CSV file containing the indices of all detected outliers is downloaded on the user's device.

Dashboard

Click experiment from dashboard **T5**

Tests: RM4 RM5

State: On dashboard

T5.1 **Action:** The user clicks on an experiment on the dashboard.
Reaction: The experiment results of the selected experiment are displayed to the user.

Create experiment

Upload CSV files **T6**

Tests: RM7 RM8

State: On create experiment page

T6.1 **Action:** User uploads a CSV file that is not corrupted.
Reaction: The CSV file is loaded to the server and its name is displayed on the create experiment page.

T6.2 **Action:** User uploads a corrupted CSV file.
Reaction: The CSV file is not loaded to the server and an error message is displayed stating that the file cannot be read.

Subspace logic **T7**

Tests: RM9 RM10

State: On create experiment page

T7.1 **Action:** The user enters a correct textual expression describing the subspace logic.
Reaction: The subspace logic is saved and used once the experiment starts running.

T7.2 **Action:** The user enters an incorrect textual expression for the subspace logic.
Reaction: An error message is displayed stating that the subspace logic expression could not be parsed.

Select ODM from dropdown menu **T8**

Tests: RM12

T8.1 **State:** On create experiment page
Action: The user selects an ODM from a dropdown menu.
Reaction: The correct ODM is selected.

Enter hyperparameters **T9**

Tests: RM11

State: On create experiment page with already selected ODM.

T9.1 **Action:** The user enters correct values for the hyperparameters.
Reaction: The parameters are saved and used once the experiment starts running.

T9.2 **Action:** The user enters incorrect values for the hyperparameters.
Reaction: An error message is displayed stating which hyperparameters are incorrect.

Try to run experiment with missing inputs **T10**

Tests: RM6 RM7 RM8 RM9 RM10 RM11 RM12

T10.1 **State:** On create experiment page with some inputs incorrect or missing.
Action: The user tries to run the experiment.
Reaction: The experiment is not run and the incorrect and missing inputs are highlighted.

Run experiment

Run experiment **T11**

Tests:

T11.1 **State:** On create experiment page with all required inputs entered correctly.
Action: The user clicks on a button to run the experiment.
Reaction: The inserted data is fed to the selected ODM which is run on the server.

Experiment results

Display results of experiment correctly **T12**

Tests: RM14

State: An experiment has been run successfully.

T12.1 **Action:** The user navigates to experiment results
Reaction: The experiment results are displayed correctly with no information missing.

11 Development environment

Documents

We used the following software to create the software requirements documentation, design documentation, and other documents:

- Overleaf
- TexLive $\geq$ 2022

Client side development

We will be using the following software for client-side application development:

- TypeScript $\geq$ 4.9
- HTML
- CSS
- Vue.js $\geq$ 3.2.45

Server side development

We will be using the following software for server-side application development:

- PostgreSQL $\geq$ 15
- Python $\geq$ 3.11.0
- Pandas $\geq$ 1.5.2
- PyOD $\geq$ 1.0.6
- Flask $\geq$ 2.2.3

Team organization

For internal organization we will use the following tools:

- Git for version control
- GitLab for collaboration

Glossary

Data point A single entry of a dataset. 11, 30

Experiment A configurable execution of outlier detection methods on subspaces of a dataset. 3–11, 13, 15–17, 19, 23–28, 30

Flask Python web framework. 29

Git A distributed version control system. 29, 30

GitLab Web application for version control based on Git. 29

Ground-truth file CSV file containing which points of a dataset are actually outliers. 7, 11, 12, 23

Hyperparameter Parameters that an ODM requires. 13, 27

Index of a data point The number of the row in which the data point is located in the given dataset. 8, 26

Overleaf Collaborative LaTeX editor. 29, 30

pandas Python library used for data analysis. 29

PostgreSQL A database system. 29

PyOD Python library providing methods for outlier detection. 7, 12, 14, 29

Python Programming language commonly used in the field of machine learning and outlier detection. 15, 29, 30

Subspace A subspace of the complete dataset. 3, 11, 12, 17, 27, 30

Subspace logic A logic that specifies in which subspaces a point of the dataset must be identified as an outlier in order to be contained in the experiment result. 7, 11, 17, 18, 23, 27

Subspace outlier A datapoint that is an outlier in a given subspace. 1, 3, 15

TexLive A distribution of the TeX typesetting system used by Overleaf. 29

TypeScript Programming language that can be used for web development. 29

Vue.js Front-end framework that can be used for building websites. 29

Acronyms

CSS Cascading Style Sheets, a language used to describe the appearance of a website. 29

CSV Comma-separated values, a file format commonly used for storing datasets. 7, 8, 11, 12, 24, 26, 27, 30

GUI Grahpical User Interface. 9, 13, 20

HTML HyperText Markup Language, a language to describe the structure of websites. 29

ODM Outlier detection method. 3, 4, 7, 8, 12, 14, 15, 17, 23, 25, 27, 28, 30

SQL Structured Query Language, a language for accessing databases. 18