

DSD+



Test Incident Report

Corso di Ingegneria, Gestione ed Evoluzione del Software

Docente:

Andrea De Lucia

Partecipanti:

Daniele Fabiano 0522501738

Tutor:

Gilberto Recupito

Francesco Paolo D'Antuono 0522501767

Indice

Introduzione	4
Prima esecuzione	5
test_dataset.py	5
TestFileBasedDatasetManager	5
test_creation	5
test_get_available_dataset_identifiers	5
test_get_dataset	5
test_get_column_names	5
test_get_great_expectations_dataset	5
test_get_batch_request	6
test_datasmell.py	6
TestDataSmellMetadata	6
test_validate_configuration	6
TestDataSmellRegistry	6
test_creation	7
test_empty_registry	7
test_one_data_smell	7
test_two_data_smells	7
TestDataSmell	7
test_is_abstract	7
test_register_data_smell	8
test_detector.py	8
TestDetectorBuilder	8
test_creation	8
test_expectations.py	9
TestExpectations	9
test_examples_of_all_expectations	9
test_profiler.py	11
TestDataSmellAwareProfiler	11
test_profiler_testcases	11
test_registration.py	12
TestExpectationRegistration	12
test_expect_column_values_to_not_contain_missing_value_smell	12
test_expect_column_values_to_not_contain_suspect_sign_smell	12
test_expect_column_values_to_not_contain_integer_as_string_smell	12
test_expect_column_values_to_not_contain_floating_point_number_as_string_s mell	12
test_expect_column_values_to_not_contain_extreme_value_smell	12
test_expect_column_values_to_not_contain_long_data_value_smell	12
test_expect_column_values_to_not_contain_integer_as_floating_point_smell	13
test_expect_column_values_to_not_contain_duplicated_value_smell	13
test_expect_column_values_to_not_contain_casing_smell	13
tests.py	13

ViewsTest	13
test_upload_csv_file	14
test_upload_png_file	14
test_customize	14
test_result	14
test_saved_results	14
ParameterFormTest	16
test_parameter_form_inside_interval	16
test_parameter_form_outside_interval	16
test_parameter_form_inside_interval_max_inf	16
test_parameter_form_outside_interval_max_inf	16
Seconda esecuzione	17
tests.py	17
ViewsTest	17
test_saved_results	17
ComputeMetricsTest	17
test_validity_with_int_value_as_float	17
Terza esecuzione	18
Quarta esecuzione	18

Introduzione

In questo documento si andranno a rieseguire i casi di test già presenti nel progetto e poi eseguire i nuovi casi di test implementati per le CR, andando poi a documentare i risultati e la soluzione eventuale che è stata implementata nel caso i test generino una o più failure. Tutto ciò viene fatto per garantire l'affidabilità e la robustezza delle componenti implementate. Ovviamente la riesecuzione dei casi di test avverrà ad ogni modifica avvenuta.

Prima esecuzione

In questa sezione verrà descritta la prima esecuzione dei seguenti casi di test, che consente di andare a definire se il sistema è ancora funzionante. Vediamo l'esecuzione dei test case nel dettaglio.

test_dataset.py

In questo modulo ci sono 2 classi che sono state implementate per il testing. Esse hanno permesso il testing di creazione di un dataset e di alcune funzionalità associate ad essi.

TestFileBasedDatasetManager

Test di unità che ha consentito di testare la classe FileBasedDatasetManager. All'interno di essa sono presenti i seguenti casi di test.

test_creation

Questo test verifica se l'oggetto di questa specifica classe viene istanziato in maniera corretta.

test_get_available_dataset_identifiers

Questo test verifica se tutti gli identificatori che sono associati ad un dataset corrispondono ai dataset specifici.

test_get_dataset

Questo test verifica se il dataset prelevato è di tipo Dataset.TestDatasetWrapper. Test di unità che ha consentito di testare la classe DatasetWrapper. All'interno di essa sono presenti i seguenti casi di test.

test_get_column_names

Questo test verifica se, per il dataset specifico prelevato dal manager, il metodo preleva in maniera corretta le colonne specifiche del medesimo.

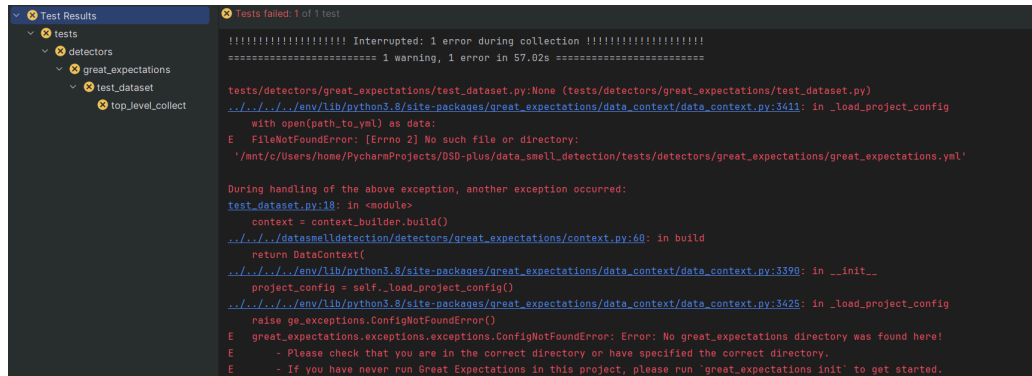
test_get_great_expectations_dataset

Questo test verifica se il dataset prelevato tramite il metodo in questione è di tipo Dataset.

test_get_batch_request

Questo test verifica se la batch request del dataset specifico è di tipo BatchRequest e ulteriori controlli sui dati specifici di questo parametro.

La riesecuzione di questi casi di test, ha portato allo scovamento di una failure come mostrato nello screen seguente.



The screenshot shows the PyCharm Test Results window. On the left, a tree view shows the test hierarchy: tests > detectors > great_expectations > test_dataset > top_level_collect. The main pane shows the test results for 'top_level_collect', which failed. The error message is as follows:

```
Tests failed: 1 of 1 test
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
===== 1 warning, 1 error in 57.02s =====

tests/detectors/great_expectations/test_dataset.py:None (tests/detectors/great_expectations/test_dataset.py)
../../../../env/lib/python3.8/site-packages/great_expectations/data_context/data_context.py:361: in _load_project_config
    with open(path_to_yaml) as data:
E   FileNotFoundError: [Errno 2] No such file or directory:
    '/mnt/c/Users/home/PycharmProjects/GSD-plus/data_smell_detection/tests/detectors/great_expectations/great_expectations.yml'

During handling of the above exception, another exception occurred:

test_dataset.py:18: in <module>
    context = context_builder.build()
../../../../datasmell/detection/detectors/great_expectations/context.py:68: in build
    return DataContext(
../../../../env/lib/python3.8/site-packages/great_expectations/data_context/data_context.py:3390: in __init__
    project_config = self._load_project_config()
../../../../env/lib/python3.8/site-packages/great_expectations/data_context/data_context.py:3425: in _load_project_config
    raise ge_exceptions.ConfigNotFoundError()
E   great_expectations.exceptions.exceptions.ConfigNotFoundError: Error: No great_expectations directory was found here!
E   - Please check that you are in the correct directory or have specified the correct directory.
E   - If you have never run Great Expectations in this project, please run 'great_expectations init' to get started.
```

test_datasmell.py

In questo modulo ci sono 3 classi che sono state implementate per il testing. Esse hanno permesso il testing di creazione di un data smell e di come esso viene registrato in Great Expectations. Per alcuni test, questo modulo si interfaccia con il modulo helper_functions.py e con il modulo fixtures.py, poiché contengono funzionalità utili al testing delle seguenti componenti. Inoltre alcuni test non sono stati implementati.

TestDataSmellMetadata

Test di unità che ha consentito di testare la classe DataSmellMetadata.

All'interno di essa sono presenti i seguenti casi di test.

test_validate_configuration

Questo test è rimasto non implementato, inoltre il metodo non è presente all'interno della classe, dunque è probabile che sia una funzionalità che non è stata più implementata o che è stata implementata in altro modo.

TestDataSmellRegistry

Test di unità che ha consentito di testare la classe DataSmellRegistry.

In dettaglio viene testato se il mapping tra un data smell e la corrispettiva aspettativa di Great Expectations viene fatta in maniera corretta.

All'interno di essa sono presenti i seguenti casi di test.

test_creation

Questo test itera i possibili valori di ProfilerDataType e si assicura che, dopo la creazione, non vengano memorizzati i data smell.

test_empty_registry

Questo test si assicura che venga restituito un dizionario vuoto dal mapping tra expectation e data smell.

test_one_data_smell

Questo test esegue la registrazione di un data smell e verifica che le chiamate al metodo `get_smell_dict_for_profiler_data_type()` restituiscano i risultati attesi. La registrazione dello smell corrispondente viene eseguita dalla fixture.

test_two_data_smells

Questo test esegue la registrazione di due data smell e garantisce che le chiamate a `get_smell_dict_for_profiler_data_type()` restituiscano i risultati attesi. La registrazione degli smell corrispondenti viene eseguita dalle fixture.

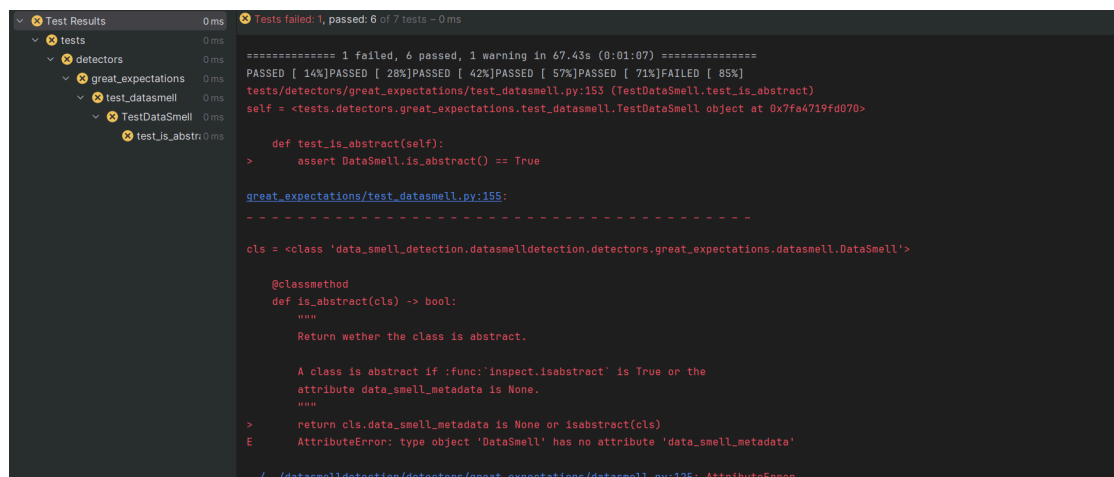
TestDataSmell

Test di unità che ha consentito di testare la classe DataSmell. All'interno di essa sono presenti i seguenti casi di test.

test_is_abstract

Questo test non è stato implementato. L'implementazione di questo test ha portato allo scovamento di un fault che ha generato una failure.

Di seguito lo screen dell'errore riscontrato.



```
Test Results 0 ms Tests failed: 1, passed: 6 of 7 tests - 0 ms
  tests 0 ms
    detectors 0 ms
      great_expectations 0 ms
        test_data_smell 0 ms
          TestDataSmell 0 ms
            test_is_abstract 0 ms

===== 1 failed, 6 passed, 1 warning in 67.43s (0:01:07) =====
PASSED [ 14%]PASSED [ 28%]PASSED [ 42%]PASSED [ 57%]PASSED [ 71%]FAILED [ 85%]
tests/detectors/great_expectations/test_data_smell.py:153 (TestDataSmell.test_is_abstract)
self = <tests.detectors.great_expectations.test_data_smell.TestDataSmell object at 0x7fa4719fd070>

    def test_is_abstract(self):
>     assert DataSmell.is_abstract() == True

great_expectations/test_data_smell.py:155:
-----
cls = <class 'data_smell_detection.data_smell_detection.detectors.great_expectations.data_smell.DataSmell'>

@classmethod
def is_abstract(cls) -> bool:
    """
    Return whether the class is abstract.

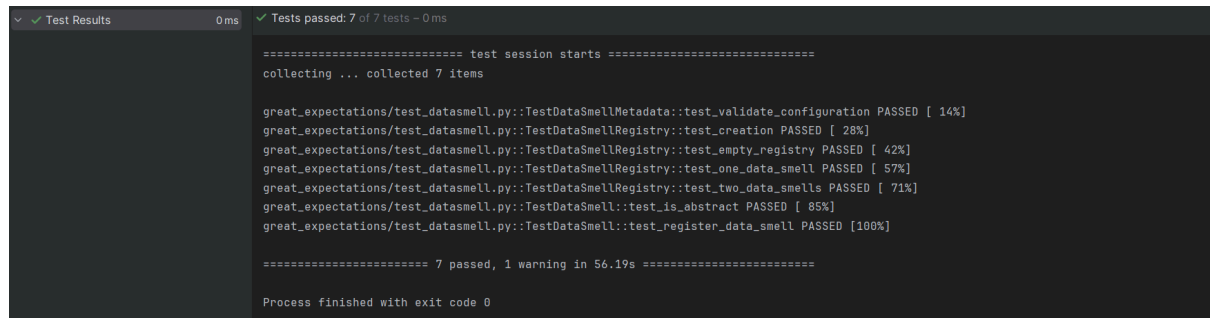
    A class is abstract if :func:`inspect.isabstract` is True or the
    attribute data_smell_metadata is None.
    """
>     return cls.data_smell_metadata is None or isabstract(cls)
E     AttributeError: type object 'DataSmell' has no attribute 'data_smell_metadata'

../../../../data_smell_detection/detectors/great_expectations/data_smell.py:125: AttributeError
```

test_register_data_smell

Questo test non è stato implementato. Ciò non è stato fatto poiché il metodo in questione viene già testato nella classe DataSmellRegistry con l'aiuto delle funzioni implementate in fixtures.py e in helper_functions.py. Dunque, la rimozione non causa particolari problemi al sistema.

Dopo aver corretto la failure scovata sul test case visto in precedenza i test sono tutti passati. Di seguito la figura che mostra i test rieseguiti e passati.



```
Test Results 0 ms Tests passed: 7 of 7 tests - 0 ms

===== test session starts =====
collecting ... collected 7 items

great_expectations/test_datasmell.py::TestDataSmellMetadata::test_validate_configuration PASSED [ 14%]
great_expectations/test_datasmell.py::TestDataSmellRegistry::test_creation PASSED [ 28%]
great_expectations/test_datasmell.py::TestDataSmellRegistry::test_empty_registry PASSED [ 42%]
great_expectations/test_datasmell.py::TestDataSmellRegistry::test_one_data_smell PASSED [ 57%]
great_expectations/test_datasmell.py::TestDataSmellRegistry::test_two_data_smells PASSED [ 71%]
great_expectations/test_datasmell.py::TestDataSmell::test_is_abstract PASSED [ 85%]
great_expectations/test_datasmell.py::TestDataSmell::test_register_data_smell PASSED [100%]

===== 7 passed, 1 warning in 56.19s =====

Process finished with exit code 0
```

test_detector.py

In questo modulo ci sono 2 classi che sono state implementate per il testing. Nel dettaglio, la classe nella quale vi sono i test case è solo una, le altre sono servite per avere accesso a funzionalità che servivano per testare quella specifica componente e per avere un mock dei dati utili per il test.

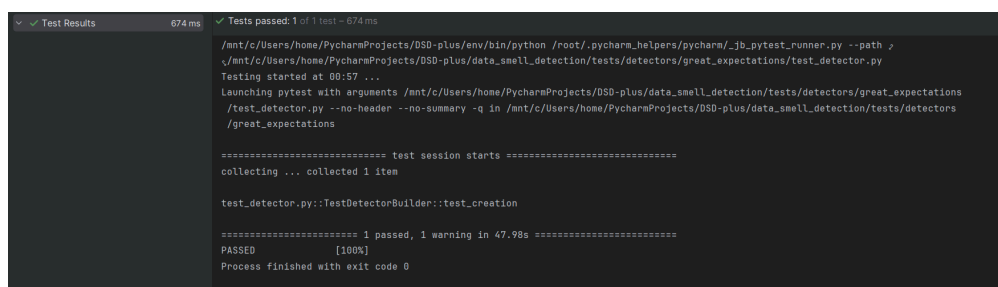
TestDetectorBuilder

Test di unità che ha consentito di testare la classe DataSmellRegistry. All'interno di essa sono presenti i seguenti casi di test.

test_creation

Questo test verifica se il build della detection con dei parametri specifici viene effettuata in maniera corretta, in questo metodo sono presenti vari casi di test che vengono raggruppati in un'unica esecuzione.

Il test risulta essere passato. La figura di seguito mostra l'esecuzione del test.



```
Test Results 674 ms Tests passed: 1 of 1 test - 674 ms

/mnt/c/Users/home/PycharmProjects/DSD-plus/env/bin/python /root/.pycharm_helpers/pycharm/_jb_pytest_runner.py --path ./
./mnt/c/Users/home/PycharmProjects/DSD-plus/data_smell_detection/tests/detectors/great_expectations/test_detector.py
Testing started at 00:57 ...
Launching pytest with arguments /mnt/c/Users/home/PycharmProjects/DSD-plus/data_smell_detection/tests/detectors/great_expectations
/test_detector.py --no-header --no-summary -q in /mnt/c/Users/home/PycharmProjects/DSD-plus/data_smell_detection/tests/detectors
/great_expectations

===== test session starts =====
collecting ... collected 1 item

test_detector.py::TestDetectorBuilder::test_creation

===== 1 passed, 1 warning in 47.98s =====
PASSED [100%]
Process finished with exit code 0
```


test_expectations.py

In questo modulo c'è un'unica classe. Essa permette di verificare se le expectations che implementano il rilevamento dei data smell, funzionano come previsto. Da notare come in questo modulo tutti i casi di test vengono eseguiti da un singolo metodo, creando una lista di test case.

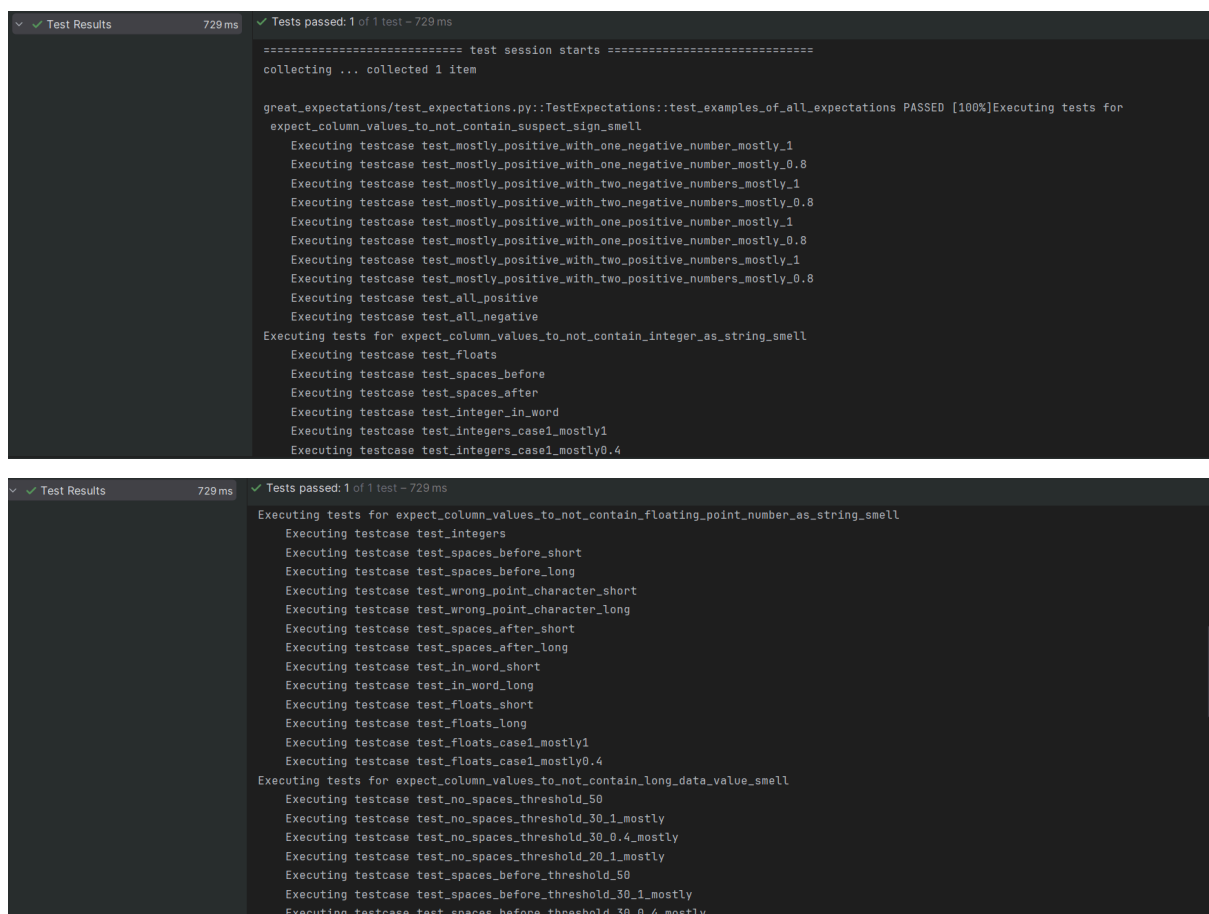
TestExpectations

Test di unità che ha consentito di testare le classi che implementano gli smell. All'interno di essa sono presenti i seguenti casi di test.

test_examples_of_all_expectations

Questo test verifica se tutti gli smell implementati hanno associato il tipo di expectation corretto.

Il test risulta essere passato. La figura di seguito mostra l'esecuzione del test.



```
Test Results 729 ms Tests passed: 1 of 1 test - 729 ms

===== test session starts =====
collecting ... collected 1 item

great_expectations/test_expectations.py::TestExpectations::test_examples_of_all_expectations PASSED [100%]
Executing tests for expect_column_values_to_not_contain_suspect_sign_smell
  Executing testcase test_mostly_positive_with_one_negative_number_mostly_1
  Executing testcase test_mostly_positive_with_one_negative_number_mostly_0.8
  Executing testcase test_mostly_positive_with_two_negative_numbers_mostly_1
  Executing testcase test_mostly_positive_with_two_negative_numbers_mostly_0.8
  Executing testcase test_mostly_positive_with_one_positive_number_mostly_1
  Executing testcase test_mostly_positive_with_one_positive_number_mostly_0.8
  Executing testcase test_mostly_positive_with_two_positive_numbers_mostly_1
  Executing testcase test_mostly_positive_with_two_positive_numbers_mostly_0.8
  Executing testcase test_all_positive
  Executing testcase test_all_negative
Executing tests for expect_column_values_to_not_contain_integer_as_string_smell
  Executing testcase test_floats
  Executing testcase test_spaces_before
  Executing testcase test_spaces_after
  Executing testcase test_integer_in_word
  Executing testcase test_integers_case1_mostly1
  Executing testcase test_integers_case1_mostly0.4

Test Results 729 ms Tests passed: 1 of 1 test - 729 ms

Executing tests for expect_column_values_to_not_contain_floating_point_number_as_string_smell
  Executing testcase test_integers
  Executing testcase test_spaces_before_short
  Executing testcase test_spaces_before_long
  Executing testcase test_wrong_point_character_short
  Executing testcase test_wrong_point_character_long
  Executing testcase test_spaces_after_short
  Executing testcase test_spaces_after_long
  Executing testcase test_in_word_short
  Executing testcase test_in_word_long
  Executing testcase test_floats_short
  Executing testcase test_floats_long
  Executing testcase test_floats_case1_mostly1
  Executing testcase test_floats_case1_mostly0.4
Executing tests for expect_column_values_to_not_contain_long_data_value_smell
  Executing testcase test_no_spaces_threshold_50
  Executing testcase test_no_spaces_threshold_30_1_mostly
  Executing testcase test_no_spaces_threshold_30_0.4_mostly
  Executing testcase test_no_spaces_threshold_20_1_mostly
  Executing testcase test_spaces_before_threshold_50
  Executing testcase test_spaces_before_threshold_30_1_mostly
  Executing testcase test_spaces_before_threshold_30_0.4_mostly
```

Test Results729 ms

Tests passed: 1 of 1 test - 729 ms

Executing testcase test_spaces_before_threshold_20_1_mostly
Executing testcase test_spaces_after_threshold_50
Executing testcase test_spaces_after_threshold_30_1_mostly
Executing testcase test_spaces_after_threshold_30_0.4_mostly
Executing testcase test_spaces_after_threshold_20_1_mostly
Executing testcase test_words_before_threshold_50
Executing testcase test_words_before_threshold_30_1_mostly
Executing testcase test_words_before_threshold_30_0.4_mostly
Executing testcase test_words_before_threshold_20_1_mostly
Executing testcase test_words_after_threshold_50
Executing testcase test_words_after_threshold_30_1_mostly
Executing testcase test_words_after_threshold_30_0.4_mostly
Executing testcase test_words_after_threshold_20_1_mostly
Executing testcase test_words_before_and_after_threshold_50
Executing testcase test_words_before_and_after_threshold_30_1_mostly
Executing testcase test_words_before_and_after_threshold_30_0.4_mostly
Executing testcase test_words_before_and_after_threshold_20_1_mostly
Executing testcase test_no_spaces_default
Executing tests for expect_column_values_to_not_contain_integer_as_floating_point_number_smell
Executing testcase test_distance0.1and0.2_eps_0.5_mostly_1
Executing testcase test_distance0.1and0.2_eps_0.25_mostly_1
Executing testcase test_distance0.1and0.2_eps_0.15_mostly_1

Test Results729 ms

Tests passed: 1 of 1 test - 729 ms

Executing tests for expect_column_values_to_not_contain_casing_smell
Executing testcase test_lowercase_only_wordcount1_mostly1
Executing testcase test_lowercase_only_wordcount2_mostly1
Executing testcase test_lowercase_only_wordcount3_mostly1
Executing testcase test_lowercase_only_wordcount3_mostly0.6
Executing testcase test_uppercase_only_wordcount1_mostly1
Executing testcase test_uppercase_only_wordcount2_mostly1
Executing testcase test_uppercase_only_wordcount3_mostly1
Executing testcase test_uppercase_only_wordcount3_mostly0.6
Executing testcase test_mixed_case_only_mostly1
Executing testcase test_mixed_case_only_mostly0.2
Executing testcase test_sentences_wordcount2_mostly1
Executing testcase test_sentences_wordcount5_mostly1
Executing testcase test_sentences_wordcount6_mostly1
Executing testcase test_sentences_wordcount6_mostly0.6
Executing testcase test_negative_wordcount1_mostly1
Executing testcase test_negative2_wordcount2_mostly1

===== 1 passed, 1 warning in 56.78s =====

test_profiler.py

In questo modulo c'è un'unica classe. Inoltre, all'interno di questo modulo, ci sono alcune funzionalità che permettono di settare i parametri dei test case specifici che si vogliono testare. Anche qui ci sono dipendenze con il modulo fixtures.py e il modulo helper_dataclasses.py. Nel dettaglio, questo modulo permette di verificare se l'applicazione dei data smell, registrati in DataSmellRegistry, alle colonne del corrispondente DataSmellType viene effettuata correttamente. L'unica classe di test implementa più test case eseguendoli con un ciclo.

TestDataSmellAwareProfiler

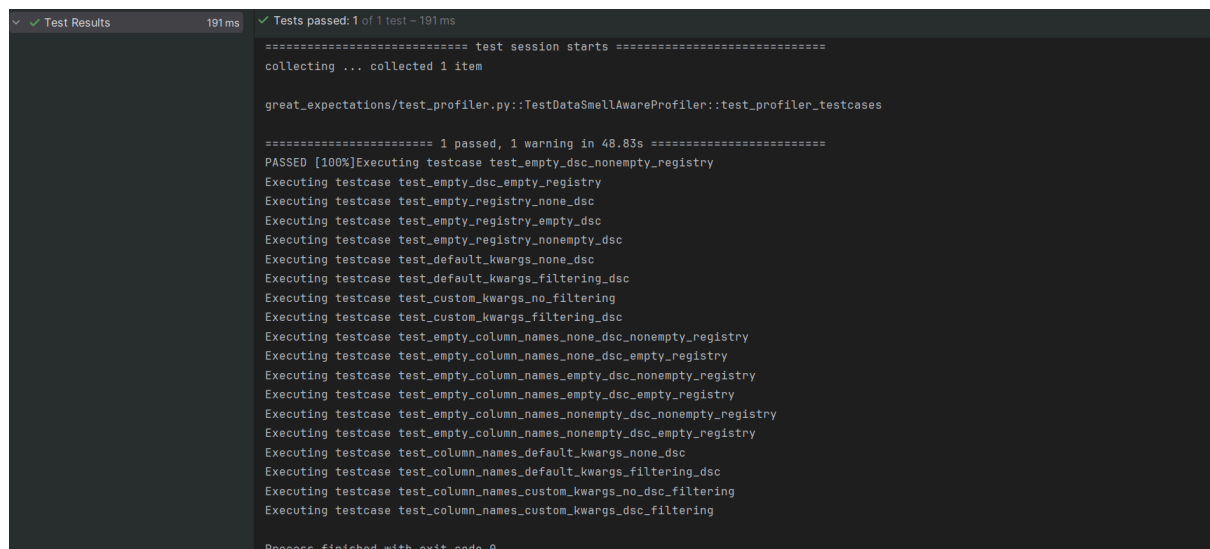
Test di unità che ha consentito di testare la classe DataSmellAwareProfiler.

All'interno di essa sono presenti i seguenti casi di test.

test_profiler_testcases

Questo test utilizza i parametri e gli oggetti di servizio definiti nello stesso modulo e verifica se il profiler lavora in maniera corretta, cioè se il legame tra data smell e colonne del dataset viene fatto correttamente.

Il test non ha rilevato failure, come è possibile vedere di seguito nella figura.



```
Test Results 191 ms ✓ Tests passed: 1 of 1 test - 191 ms
===== test session starts =====
collecting ... collected 1 item

great_expectations/test_profiler.py::TestDataSmellAwareProfiler::test_profiler_testcases

===== 1 passed, 1 warning in 48.83s =====
PASSED [100%]Executing testcase test_empty_dsc_nonempty_registry
Executing testcase test_empty_dsc_empty_registry
Executing testcase test_empty_registry_none_dsc
Executing testcase test_empty_registry_empty_dsc
Executing testcase test_empty_registry_nonempty_dsc
Executing testcase test_default_kwargs_none_dsc
Executing testcase test_default_kwargs_filtering_dsc
Executing testcase test_custom_kwargs_no_filtering
Executing testcase test_custom_kwargs_filtering_dsc
Executing testcase test_empty_column_names_none_dsc_nonempty_registry
Executing testcase test_empty_column_names_none_dsc_empty_registry
Executing testcase test_empty_column_names_empty_dsc_nonempty_registry
Executing testcase test_empty_column_names_empty_dsc_empty_registry
Executing testcase test_empty_column_names_nonempty_dsc_nonempty_registry
Executing testcase test_empty_column_names_nonempty_dsc_empty_registry
Executing testcase test_column_names_default_kwargs_none_dsc
Executing testcase test_column_names_default_kwargs_filtering_dsc
Executing testcase test_column_names_custom_kwargs_no_dsc_filtering
Executing testcase test_column_names_custom_kwargs_dsc_filtering

Process finished with exit code 0
```

test_registration.py

In questo modulo c'è una singola classe. Esso permette di verificare che l'importazione del modulo `datasmelldetection.detectors.great_expectations` registri le expectation corrispondenti per l'individuazione dei data smell.

TestExpectationRegistration

Test di unità che ha consentito di testare le classi

`ExpectValuesToNotContain<SmellType>` (<SmellType> rappresenta la variazione di questa parte del nome della classe, per ogni smell).

All'interno di essa sono presenti i seguenti casi di test.

`test_expect_column_values_to_not_contain_missing_value_smell`

Questo test verifica se la detection del missing value smell viene svolto in maniera corretta, andando a confrontare il `DataSmellRegistry` specifico.

`test_expect_column_values_to_not_contain_suspect_sign_smell`

Questo test verifica se la detection del Suspect Sign smell viene svolto in maniera corretta, andando a confrontare il `DataSmellRegistry` specifico.

`test_expect_column_values_to_not_contain_integer_as_string_smell`

Questo test verifica se la detection del Integer As String Smell viene svolto in maniera corretta, andando a confrontare il `DataSmellRegistry` specifico.

`test_expect_column_values_to_not_contain_floating_point_number_as_string_smell`

Questo test verifica se la detection del Floating Point Number As String smell viene svolto in maniera corretta, andando a confrontare il `DataSmellRegistry` specifico.

`test_expect_column_values_to_not_contain_extreme_value_smell`

Questo test verifica se la detection del Extreme Value smell viene svolto in maniera corretta, andando a confrontare il `DataSmellRegistry` specifico.

`test_expect_column_values_to_not_contain_long_data_value_smell`

Questo test verifica se la detection del Long Data Value smell viene svolto in maniera corretta, andando a confrontare il `DataSmellRegistry` specifico.

test_expect_column_values_to_not_contain_integer_as_floating_point_smell

Questo test verifica se la detection del Integer As Floating Point smell viene svolto in maniera corretta, andando a confrontare il DataSmellRegistry specifico.

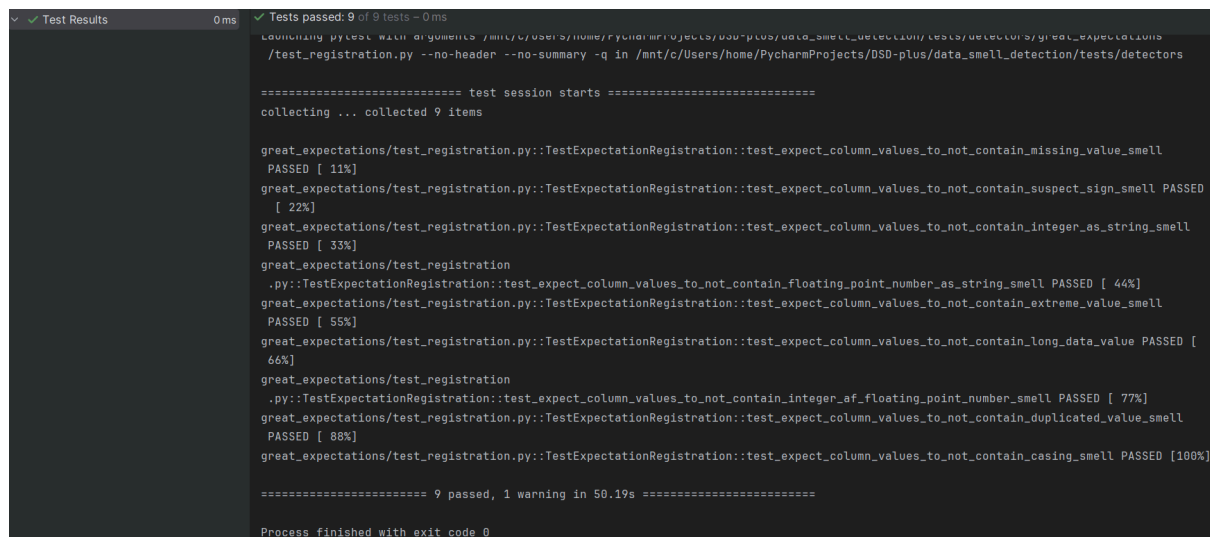
test_expect_column_values_to_not_contain_duplicated_value_smell

Questo test verifica se la detection del Duplicated Value smell viene svolto in maniera corretta, andando a confrontare il DataSmellRegistry specifico.

test_expect_column_values_to_not_contain_casing_smell

Questo test verifica se la detection del Casing smell viene svolto in maniera corretta, andando a confrontare il DataSmellRegistry specifico.

I test sono tutti passati. Di seguito lo screen che mostra l'esecuzione dei test case.



```
Test Results 0ms ✓ Tests passed: 9 of 9 tests - 0ms
Launching pytest with arguments /mnt/c/Users/home/PycharmProjects/DSD-plus/data_smell_detection/tests/detectors/great_expectations
/test_registration.py --no-header --no-summary -q in /mnt/c/Users/home/PycharmProjects/DSD-plus/data_smell_detection/tests/detectors

===== test session starts =====
collecting ... collected 9 items

great_expectations/test_registration.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_missing_value_smell
PASSED [ 11%]
great_expectations/test_registration.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_suspect_sign_smell PASSED
[ 22%]
great_expectations/test_registration.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_integer_as_string_smell
PASSED [ 33%]
great_expectations/test_registration
.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_floating_point_number_as_string_smell PASSED [ 44%]
great_expectations/test_registration.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_extreme_value_smell
PASSED [ 55%]
great_expectations/test_registration.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_long_data_value PASSED [
66%]
great_expectations/test_registration
.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_integer_of_floating_point_number_smell PASSED [ 77%]
great_expectations/test_registration.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_duplicated_value_smell
PASSED [ 88%]
great_expectations/test_registration.py::TestExpectationRegistration::test_expect_column_values_to_not_contain_casing_smell PASSED [100%]

===== 9 passed, 1 warning in 50.19s =====

Process finished with exit code 0
```

tests.py

In questo modulo ci sono 2 classi che testano la web_application e verificano il funzionamento corretto delle funzionalità del sistema. Per eseguire questi casi di test, il database deve essere svuotato altrimenti da problemi il salvataggio dei dati. Sono stati corretti vari errori di codice, tra cui un errore di ritorno che non permetteva di avere a disposizione il dizionario context, che possiede al suo interno tutte le informazioni, le quali venivano computate dalle funzionalità in views.py.

ViewsTest

Test di unità che consente di testare le funzionalità principali del sistema all'interno del modulo views.py.

All'interno di essa sono presenti i seguenti casi di test.

test_upload_csv_file

Questo test consente di verificare il corretto caricamento nel sistema del file .csv che verrà utilizzato per la detection.

test_upload_png_file

Questo test consente di verificare se il sistema restituisce il messaggio di errore, quando l'utente inserisce un tipo di file diverso dal formato .csv.

test_customize

Questo test consente di verificare se la personalizzazione della detection funziona in maniera corretta.

test_result

Questo test consente di verificare se i risultati sono restituiti correttamente all'utente.

test_saved_results

Questo test consente di verificare se i risultati, per l'utente registrato, vengono salvati in maniera corretta.

I test non sono passati, come è possibile osservare dalle figure che sono state fornite di seguito.

```
Test Results 19 sec 126 ms Tests failed: 5, passed: 4 of 9 tests - 19 sec 126 ms
  app 19 sec 126 ms
    tests 19 sec 126 ms
      ViewsTest 17 sec 743 ms
        test_custo 15 sec 340 ms
        test_result 641 ms
        test_saved_resul 492 ms
        test_upload_csv 656 ms
        test_upload_png 614 ms

/mnt/c/Users/home/PycharmProjects/DSD-plus/env/bin/python /root/.pycharm_helpers/pycharm/_jb_pytest_runner.py --path ,
  \mnt/c/Users/home/PycharmProjects/DSD-plus/web_application/argon-dashboard-django/app/tests.py
Testing started at 16:33 ...
Launching pytest with arguments /mnt/c/Users/home/PycharmProjects/DSD-plus/web_application/argon-dashboard-django/app/tests.py
--no-header --no-summary -q in /mnt/c/Users/home/PycharmProjects/DSD-plus/web_application/argon-dashboard-django

===== test session starts =====
collecting ... collected 9 items

app/tests.py::ViewsTest::test_customize
app/tests.py::ViewsTest::test_result
app/tests.py::ViewsTest::test_saved_results
app/tests.py::ViewsTest::test_upload_csv_file
app/tests.py::ViewsTest::test_upload_png_file
app/tests.py::ParameterFormTest::test_parameter_form_inside_interval
app/tests.py::ParameterFormTest::test_parameter_form_inside_interval_max_inf
app/tests.py::ParameterFormTest::test_parameter_form_outside_interval
app/tests.py::ParameterFormTest::test_parameter_form_outside_interval_max_inf

===== 5 failed, 4 passed, 13 warnings in 37.62s =====
```

```
Test Results 23 sec 699 ms Tests failed: 5, passed: 4 of 9 tests - 23 sec 699 ms
  app 23 sec 699 ms
    tests 23 sec 699 ms
      ViewsTest 22 sec 282 ms
        test_custo 19 sec 667 ms
        test_result 542 ms
        test_saved_resul 597 ms
        test_upload_csv 768 ms
        test_upload_png 708 ms

FAILED [ 11%]
app/tests.py:115 (ViewsTest.test_customize)
self = <app.tests.ViewsTest testMethod=test_customize>

def test_customize(self):
    response = self.client.get(reverse('customize'))
    > self.assertEqual(response.context['forms']['Believability Smells'][DataSmellType('Duplicated Value Smell')]['checkbox'],
'smell_checked')
E   TypeError: 'NoneType' object is not subscriptable

app/tests.py:118: TypeError
FAILED [ 22%]
app/tests.py:140 (ViewsTest.test_result)
self = <app.tests.ViewsTest testMethod=test_result>

def test_result(self):
    response = self.client.get(reverse('result'))
    > self.assertEqual(response.context['column_names'], [self.column1.column_name, self.column2.column_name])
E   TypeError: 'NoneType' object is not subscriptable
```

```
Test Results 23 sec 699 ms Tests failed: 5, passed: 4 of 9 tests - 23 sec 699 ms
app 23 sec 699 ms
tests 23 sec 699 ms
  ViewsTest 22 sec 282 ms
    test_custo 19 sec 667 ms
    test_result 542 ms
    test_saved_resul 597 ms
    test_upload_csv_ 768 ms
    test_upload_png_ 708 ms

app/tests.py:149 (ViewsTest.test_saved_results)
self = <app.tests.ViewsTest testMethod=test_saved_results>

def test_saved_results(self):
    self.detected_smell = DetectedSmell.objects.create(data_smell_type=self.smell_type1, total_element_count=200,
        faulty_element_count=10, faulty_list=["hi", "jo", "ho"], belonging_column=self.column1)
    response = self.client.get(reverse('saved'))
    > self.assertEqual(response.context['results'], {'Titanic.csv': {self.column1: [self.detected_smell], self.column2: []}})
    E TypeError: 'NoneType' object is not subscriptable

app/tests.py:153: TypeError
FAILED [ 44%]
app/tests.py:93 (ViewsTest.test_upload_csv_file)
self = <app.tests.ViewsTest testMethod=test_upload_csv_file>

def test_upload_csv_file(self):
    response = self.client.get(reverse('upload'))

    with open(SMELL_FOLDER+'test.csv') as csv_file:
        request = self.client.post(reverse('upload'), {'upload': csv_file})

    > self.assertEqual(request.context.get('message'), None)
    E AttributeError: 'NoneType' object has no attribute 'get'

app/tests.py:100: AttributeError
```

```
Test Results 19 sec 177 ms Tests failed: 5, passed: 4 of 9 tests - 19 sec 177 ms
app 19 sec 177 ms
tests 19 sec 177 ms
  ViewsTest 17 sec 804 ms
    test_custo 15 sec 784 ms
    test_result 785 ms
    test_saved_resul 344 ms
    test_upload_csv_ 461 ms
    test_upload_png_ 450 ms

app/tests.py:100:
-----
../env/lib/python3.8/site-packages/django/test/testcases.py:859: in assertTemplateUsed
    self._assert_template_used(template_name, template_names, msg_prefix, count)
../env/lib/python3.8/site-packages/django/test/testcases.py:823: in _assert_template_used
    self.fail(msg_prefix + "No templates used to render the response")
E AssertionError: No templates used to render the response
FAILED [ 55%]
app/tests.py:101 (ViewsTest.test_upload_png_file)
self = <app.tests.ViewsTest testMethod=test_upload_png_file>

def test_upload_png_file(self):

    with open(SMELL_FOLDER+'test.png') as csv_file:
        response = self.client.post(reverse('upload'), {'upload': csv_file}, content_type='text/csv', follow=True)

    > self.assertEqual(response.context['message'], 'Upload a .csv file.')
    E TypeError: 'NoneType' object is not subscriptable

app/tests.py:107: TypeError
PASSED [ 66%]PASSED [ 77%]PASSED [ 88%]PASSED [100%]
Process finished with exit code 1
```

ParameterFormTest

Test che consente di verificare se i parametri che vengono utilizzati per la customization sono corretti e rientrano nel range di valori ammesso.

All'interno di essa sono presenti i seguenti casi di test. Questi test sono stati eseguiti in maniera corretta non appena è stato settato l'environment in maniera corretta e svuotato il db.

test_parameter_form_inside_interval

Questo test verifica se i parametri rispettano l'intervallo di valori predefinito.

test_parameter_form_outside_interval

Questo test verifica se i parametri non rispettano l'intervallo di valori predefinito.

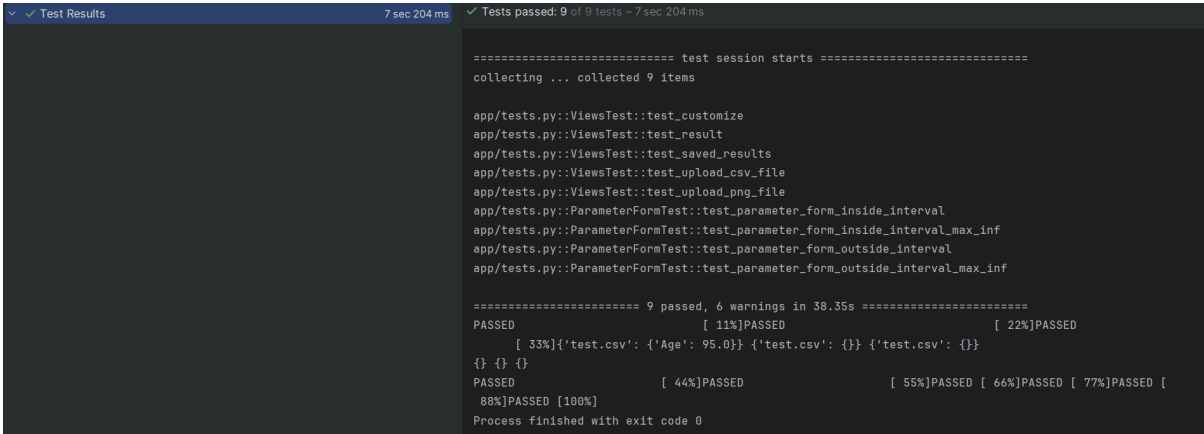
test_parameter_form_inside_interval_max_inf

Questo test verifica se i parametri rispettano l'intervallo di valori predefinito ai bordi dei possibili input.

test_parameter_form_outside_interval_max_inf

Questo test verifica se i parametri non rispettano l'intervallo di valori predefinito ai bordi dei possibili input.

Di seguito la figura che mostra i test passati.



```
Test Results 7 sec 204 ms Tests passed: 9 of 9 tests - 7 sec 204 ms

===== test session starts =====
collecting ... collected 9 items

app/tests.py::ViewsTest::test_customize
app/tests.py::ViewsTest::test_result
app/tests.py::ViewsTest::test_saved_results
app/tests.py::ViewsTest::test_upload_csv_file
app/tests.py::ViewsTest::test_upload_png_file
app/tests.py::ParameterFormTest::test_parameter_form_inside_interval
app/tests.py::ParameterFormTest::test_parameter_form_inside_interval_max_inf
app/tests.py::ParameterFormTest::test_parameter_form_outside_interval
app/tests.py::ParameterFormTest::test_parameter_form_outside_interval_max_inf

===== 9 passed, 6 warnings in 38.35s =====
PASSED [ 11%]PASSED [ 22%]PASSED [ 33%]{'test.csv': {'Age': 95.0}} {'test.csv': {}} {'test.csv': {}}
{} {} {}
PASSED [ 44%]PASSED [ 55%]PASSED [ 66%]PASSED [ 77%]PASSED [ 88%]PASSED [100%]
Process finished with exit code 0
```


Seconda esecuzione

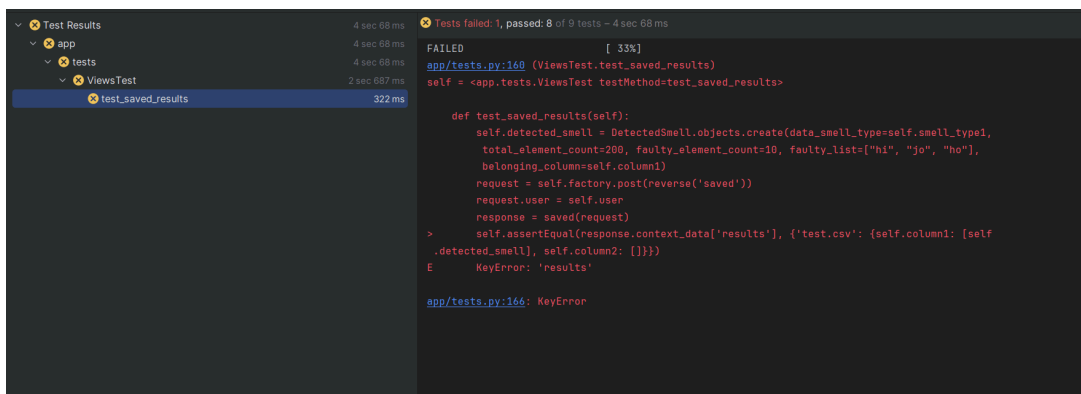
Qui viene documentata la seconda esecuzione dei casi di test, evidenziando solo i casi di test che hanno generato un failure dopo che sono state inserite le nuove componenti, che hanno permesso di implementare la CR_02, che riguarda il calcolo delle metriche, le quali serviranno per il nuovo meccanismo di reporting. Di seguito vengono mostrati i casi di test che hanno generato failure.

tests.py

ViewsTest

test_saved_results

Il seguente test case ha generato una failure.
Ecco la figura che mostra l'errore generato.



The screenshot shows a test runner interface with a sidebar on the left and a main panel on the right. The sidebar lists the test hierarchy: Test Results (4 sec 68 ms), app (4 sec 68 ms), tests (4 sec 68 ms), ViewsTest (2 sec 687 ms), and test_saved_results (322 ms). The main panel displays the failure details for the test_saved_results test case. It shows the test method name, the test code, and the error message: `KeyError: 'results'`.

```
Test Results 4 sec 68 ms
  app 4 sec 68 ms
    tests 4 sec 68 ms
      ViewsTest 2 sec 687 ms
        test_saved_results 322 ms

Tests failed: 1, passed: 8 of 9 tests - 4 sec 68 ms
FAILED [ 33%]
app/tests.py:160 (ViewsTest.test_saved_results)
self = <app.tests.ViewsTest testMethod=test_saved_results>

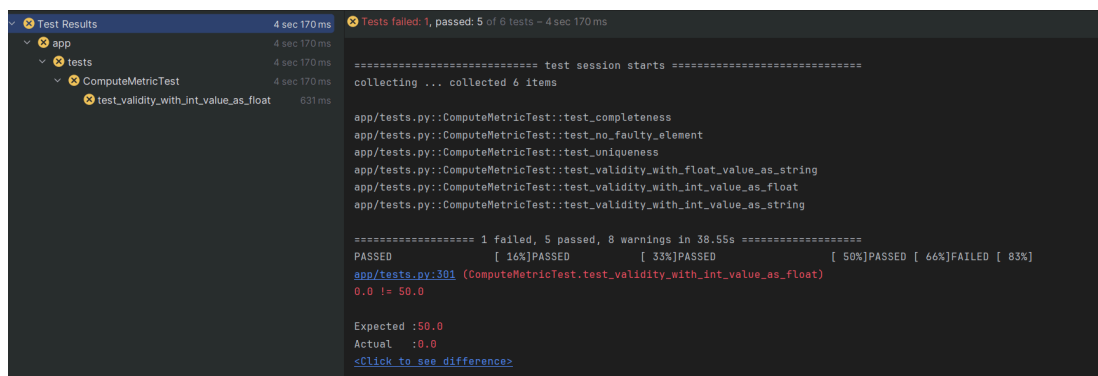
def test_saved_results(self):
    self.detected_smell = DetectedSmell.objects.create(data_smell_type=self.smell_type1,
    total_element_count=200, faulty_element_count=10, faulty_list=["hi", "jo", "ho"],
    belonging_column=self.column1)
    request = self.factory.post(reverse('saved'))
    request.user = self.user
    response = saved(request)
    > self.assertEqual(response.context_data['results'], {'test.csv': {self.column1: [self
    .detected_smell], self.column2: []}})
E   KeyError: 'results'

app/tests.py:166: KeyError
```

ComputeMetricsTest

test_validity_with_int_value_as_float

Il seguente test case ha generato una failure causata da come viene processato il dataset dal sistema. Nella figura seguente viene mostrato l'errore generato.



The screenshot shows a test runner interface with a sidebar on the left and a main panel on the right. The sidebar lists the test hierarchy: Test Results (4 sec 170 ms), app (4 sec 170 ms), tests (4 sec 170 ms), ComputeMetricTest (4 sec 170 ms), and test_validity_with_int_value_as_float (631 ms). The main panel displays the failure details for the test_validity_with_int_value_as_float test case. It shows the test method name, the test code, and the error message: `AssertionError: 0.0 != 50.0`.

```
Test Results 4 sec 170 ms
  app 4 sec 170 ms
    tests 4 sec 170 ms
      ComputeMetricTest 4 sec 170 ms
        test_validity_with_int_value_as_float 631 ms

Tests failed: 1, passed: 5 of 6 tests - 4 sec 170 ms
===== test session starts =====
collecting ... collected 6 items

app/tests.py::ComputeMetricTest::test_completeness
app/tests.py::ComputeMetricTest::test_no_faulty_element
app/tests.py::ComputeMetricTest::test_uniqueness
app/tests.py::ComputeMetricTest::test_validity_with_float_value_as_string
app/tests.py::ComputeMetricTest::test_validity_with_int_value_as_float
app/tests.py::ComputeMetricTest::test_validity_with_int_value_as_string

===== 1 failed, 5 passed, 8 warnings in 38.55s =====
PASSED [ 16%]PASSED [ 33%]PASSED [ 50%]PASSED [ 66%]FAILED [ 83%]
app/tests.py:301 (ComputeMetricTest.test_validity_with_int_value_as_float)
0.0 != 50.0

Expected :50.0
Actual   :0.0
<Click to see difference>
```

Terza esecuzione

La terza esecuzione dei casi di test, ha previsto l'esecuzione dei nuovi casi di test realizzati per la CR_01, ovvero l'implementazione delle nuove detection di data smell. In questa terza esecuzione, nessun caso di test ha generato failure.

Quarta esecuzione

La quarta esecuzione dei casi di test, ha previsto l'esecuzione dei nuovi casi di test realizzati per la CR_03, ovvero l'estensione del reporting system. In questa quarta esecuzione, un caso di test ha generato una failure. Questa problematica si è diffusa su tutti i casi di test successivi ma è stata inizialmente riscontrata sul caso di test *test_no_csv_file*. Di seguito viene mostrato l'errore generato.

