DSD+

Test Plan

Corso di Ingegneria, Gestione ed Evoluzione del Software

Docente:

Andrea De Lucia

Tutor:

Gilberto Recupito

Partecipanti:

Daniele Fabiano 0522501738

Francesco Paolo D'Antuono 0522501767

Indice

Introduzione

In questo documento si andranno ad esplicitare tutte le strategie e le attività che riguardano la fase di testing. Inoltre si individueranno i requisiti e le componenti che si andranno a testare, non prima di aver effettuato un'analisi approfondita dei casi di test già implementati.

System overview

In questa sezione si fa riferimento ai casi di test attuali più significativi del sistema. Ciò viene fatto per garantire la tracciabilità di quello che è stato già fatto e derivare una possibile strategia di testing, utilizzata nello sviluppo dei casi di test. Da notare che, per garantire un riuso del codice, sono stati realizzati tre moduli di supporto, che consentono di avere dei mock già pronti per effettuare determinati casi di test. I moduli appena descritti sono:

- fixtures.py: In questo modulo sono inizializzate le risorse necessarie alla fase di testing, che sono successivamente utilizzate dai rispettivi casi di test per definire lo stato e l'ambiente di esecuzione per il corretto funzionamento del test.
- helper_dataclasses.py: Questa classe di dati è pensata principalmente per memorizzare informazioni sui data smell al fine di testarne la presenza nei data smell registry o nelle suite di expectations.
- helper_functions.py: In questo modulo vengono definite delle funzioni di supporto, che permettono anche di effettuare dei controlli specifici, i quali sono stati utili per i test case che sono stati poi implementati. Inoltre in alcuni casi, le funzioni messe a disposizione sono utilizzate come driver principali per l'esecuzione dei casi di test.

L'esecuzione di tutti i casi di test (non solo quelli presentati qui) viene documentata all'interno del "Test Summary Report". Qui è possibile visualizzare i casi di test esistenti

Features to be tested/not to be tested

In questa sezione si andranno a mostrare le funzionalità del sistema che devono essere testate. Ciò che si andrà a testare, saranno le nuove funzionalità che derivano dalle change requests, definite nel documento denominato “Maintenance Report”. Questi test serviranno per garantire la corretta implementazione di queste funzionalità. Dunque le funzionalità che verranno testate sono:

- **CR_01**: Si testeranno le classi che saranno inserite per la detection dei nuovi data smell.
- **CR_02**: Si testeranno i metodi che saranno inseriti per il calcolo delle metriche.
- **CR_03**: Si testerà il sistema, per verificare se ciò che è stato inserito nel nuovo meccanismo di reporting corrisponde ai valori attesi e se le informazioni vengono visualizzate in maniera corretta.

Nella sezione “Test Cases” verranno esplicitati i test case, che saranno poi specificati in dettaglio nel documento di “Test Case Specification”.

Pass/Fail criteria

Il test che stiamo per eseguire, mira a rilevare eventuali difetti nel sistema, che saranno successivamente corretti. Ogni test include un oracolo, che indica l'output previsto per un determinato input. Un test sarà considerato superato se, dato un certo input dell'oracolo, l'output prodotto dal sistema coincide con quello indicato dall'oracolo. Diversamente, se con un dato input dell'oracolo, l'output del sistema non coincide con quello previsto, il test sarà considerato non superato.

Approccio

In questa fase si descrive l'approccio generale che andremo ad adottare per il testing. Ciò ci permetterà di capire che tipo di test dovrà essere costruito e in quale momento farlo.

Regression Testing

La prima attività che si svolgerà per la fase di testing è il regression testing. Questa attività è fondamentale per determinare la robustezza dei casi di test già presenti e se c'è bisogno di completare la test suite, nel caso in cui il testing delle componenti non è stato completato per mancanza di tempo o per necessità. Il regression testing verrà documentato in uno specifico documento, denominato Test Summary Report dove, non solo verrà documentata la riesecuzione dei casi di test, ma verrà anche descritto il tipo di failure che genera il caso di test (nel caso in cui il test non passa) e come il problema è stato risolto. Inoltre, questa attività verrà rieseguita non appena la CR su cui si sta lavorando, verrà implementata.

Unit Testing

In questa fase verranno costruiti i test di unità, cioè i test che andranno a verificare il funzionamento in maniera corretta delle singole componenti implementate. La tecnica che verrà utilizzata per il testing è la tecnica black-box. Di seguito, verranno descritte le partizioni di input definite, per le funzionalità da testare.

CR_01

Per questa change request sono state definite due classi da testare che sono:

- **ExpectColumnValuesToNotContainSpacingSmell**.
- **ExpectColumnValuesToNotContainSpecialCharacterSmell**.

In questo caso l'input per il testing di queste classi risulta essere la colonna di un dataset. Dunque, è stata utilizzata la tecnica di partizione dell'input per il testing, denominata *Weak Equivalence Class Testing*. Le motivazioni dietro questa scelta sono dovute al fatto che i possibili input sono descrivibili da un basso numero di proprietà, il che ci permette in maniera abbastanza immediata di dividere l'input in partizioni specifiche. Inoltre, dato che il tipo di input che abbiamo a disposizione, risulta essere una colonna del dataset, non vi è la necessità di avere una partizione dell'input così particolarmente dettagliata e i pochi casi rappresentativi che verranno scelti, sulla base delle caratteristiche da testare, permetteranno di fornire un buon

livello di copertura di casi di test. Le partizioni si baseranno sul numero di fault presenti in una colonna e il tipo di smell da rilevare.

Sulla base delle partizioni individuate, saranno definiti i casi di test.

Le classi di input considerano gli insiemi

- **A**: {Insieme dei possibili valori faulty degli input sulle colonne};
- **B**: {Insieme dei possibili valori di input sugli elementi singoli di una colonna}

che sono partizionati come segue:

- **A1**: {Nessun elemento faulty individuato in una colonna}.
- **A2**: {Un elemento faulty individuato in una colonna}.
- **A3**: {Più di un elemento faulty individuato in una colonna}.
- **B1**: {Elemento che contiene una parola vuota, una singola parola o più parole con un singolo carattere di spacing che le separa}.
- **B2**: {Elemento nel quale si ha una parola con uno o più spazi iniziali}.
- **B3**: {Elemento nel quale si ha una parola con uno o più spazi alla fine}.
- **B4**: {Elemento nel quale si hanno due o più parole con più di uno spazio che le separa}.
- **B5**: {Elemento che è di tipo alfanumerico}.
- **B6**: {Elemento che contiene caratteri non di tipo alfanumerico}.

Esse sono basate sul concetto di elemento *faulty*, cioè un elemento di una colonna che viene individuato dal tool come smell.

I casi di test che sono stati derivati, sono elencati nella tabella qui sotto.

Test case	A	B
TC_01_CR_01	A1	B1
TC_02_CR_01	A1	B5
TC_03_CR_01	A2	B2
TC_04_CR_01	A3	B3
TC_05_CR_01	A2	B4
TC_06_CR_01	A3	B6

CR_02

Per questa change request è stata definita una funzione nel modulo *manage_metrics.py*, denominata **compute_metric**, che restituisce i valori globali e locali di Completeness, Uniqueness e Validity, a seconda della metrica considerata (dettagli nel documento di Maintenance nella parte di Object Design).

Anche in questo abbiamo utilizzato il *Weak Equivalence Class Testing*, per gli stessi motivi spiegati in precedenza. Il nostro input sarà una colonna con dieci elementi.

Le classi di input considerano gli insiemi

- **C**: {Insieme dei possibili valori faulty degli input sulle colonne};
- **D**: {Insieme dei possibili valori di input sugli elementi singoli di una colonna}

che sono partizionati come segue:

- **C1**: {Nessun elemento faulty individuato in una colonna}.
- **C2**: {Uno o più elementi faulty individuati in una colonna}.
- **D1**: {Elemento che non contiene valore}.
- **D2**: {Elemento che è un duplicato}.
- **D3**: {Elemento che contiene valore intero ma formattato come stringa}.
- **D4**: {Elemento che contiene valore intero ma formattato come numero decimale}.
- **D5**: {Elemento che contiene valore decimale ma formattato come stringa}.
- **D6**: {Elemento che contiene un valore non duplicato, non vuoto e che il tipo di dato con cui viene riconosciuto è coerente con il valore considerato}.

I casi di test che sono stati derivati, sono elencati nella tabella qui sotto.

Test case	A	B
TC_01_CR_02	C1	D6
TC_02_CR_02	C2	D1
TC_03_CR_02	C2	D2
TC_04_CR_02	C2	D3
TC_05_CR_02	C2	D4
TC_06_CR_02	C2	D5

System Testing

In questa fase verrà effettuato il testing di sistema, per verificare che le singole componenti siano state correttamente integrate all'interno del sistema e per determinare se ci sono ulteriori problemi, dovuti dalle modifiche apportate.

Per questo tipo di testing, gli input saranno dei dataset che rappresentano contesti differenti e quindi, saranno composti da tipi di dati diversi, in modo tale da verificare e generalizzare il corretto comportamento delle funzionalità del sistema.

Di seguito sono elencati i casi di test:

Test case	dataset
TC_01_CR_03	sf-salaries.csv
TC_02_CR_03	red-light-camera-violations.csv
TC_03_CR_03	metropolitan-objects.csv
TC_04_CR_03	indicators-by-company.csv
TC_05_CR_03	food-inspections.csv
TC_06_CR_03	aws-cyberattacks.csv
TC_07_CR_03	no-smell-dataset.csv
TC_08_CR_03	empty-dataset.csv
TC_09_CR_03	empty-column-dataset.csv
TC_10_CR_03	no-csv-file.png

Sospensione e Ripresa

Criteri di sospensione

Il testing deve essere portato avanti, finché non sono state testate tutte le feature che sono state definite in precedenza. In alcuni casi, il testing può essere sospeso, in particolare quando viene trovato un failure oppure quando vi è il bisogno di dedicarsi ad una nuova implementazione. Ciò viene fatto per evitare di proseguire con test, che in realtà hanno già avuto successo e quindi risolvere il prima possibile il problema riscontrato.

Criteri di ripresa

Il testing verrà ripreso non appena verranno risolte tutte le failure riscontrate in precedenza, oppure quando la nuova feature è stata considerata implementata al completo. Ovviamente sarà un processo iterativo, che porterà a varie sospensioni e riprese, finché tutti i problemi scovati dai test non verranno risolti e tutte le feature non sono state implementate.

Materiali di Testing

Per riuscire a facilitare il testing delle componenti, il processo è stato automatizzato attraverso pytest (l'equivalente di JUnit in java) che consente di eseguire i casi di test in maniera semplice e veloce. Inoltre, per il system testing, è stato utilizzato Selenium IDE, un browser tool che ci ha consentito di testare il sistema e verificare il corretto comportamento delle funzionalità implementate, controllando anche se sono state integrate con le componenti già esistenti in maniera corretta.

Replica di Esecuzione dei Test Cases

In questa sezione verrà descritto come replicare l'esecuzione dei casi di test, per chi vorrà in futuro utilizzarli e/o modificarli.

CR_01 Test Cases

Per questi casi di test bisogna eseguire i seguenti passi:

1. Posizionarsi nella cartella *tests/detectors/great_expectations* del package *data_smell_detection;*
2. Eseguire il modulo test_expectations.py.

CR_02 Test Cases

Per questi casi di test bisogna eseguire i seguenti passi:

1. Posizionarsi nella cartella *argon_dashboard_django/app* del package *web_application;*
2. Copiare i file .csv che rappresentano gli input dei test case dalla cartella *test_sets* e il file *test.csv* e incollarli nella cartella *../core/media* (cartella di riferimento alla quale accede Great Expectations per i data source);
3. Nella run configuration delle suite di test impostare la working directory fino ad *argon_dashboard_django/;*
4. Eseguire il modulo tests.py.

NB: Ogni volta che si esegue questo modulo bisogna reinserire il file *test.csv* nella cartella *../core/media,* poiché viene cancellato da un test che verifica il corretto funzionamento della cancellazione di un file.

CR_03 Test Cases

Per questi casi di test bisogna eseguire i seguenti passi:

1. Installare l'estensione di Selenium IDE sui browser supportati dal tool;
2. Avviare il server del progetto;
3. Assicurarsi che la sezione relativa ai risultati sia vuota (si consiglia di svuotare il db);
4. Caricare il file .side all'interno del tool;
5. Eseguire i casi di test in un ordine specifico, per evitare errori di esecuzione inattesi.

NB: Nel caso si deve effettuare una riesecuzione della suite, assicurarsi che i file non siano più presenti all'interno della cartella *web_application/argon_dashboard_django_core_media* dove vengono salvati tutti i file analizzati.

L'ordine di esecuzione è il seguente:

1. test_no_csv_file;
2. test_empty_dataset;
3. test_empty_column_dataset;
4. test_no_smell_dataset;
5. test_aws_cyberattacks;
6. test_food_inspections;
7. test_indicators_by_company;
8. test_metropolitan_objects;
9. test_red_light;
10. test_salaries;

Test Cases

In questa sezione definiremo i casi di test, che saranno poi specificati nel dettaglio all'interno del documento "Test Case Specification".

CR_01 Test Cases

Nome	ID
no_spacing_smell	TC_01_CR_01
no_special_character_smell	TC_02_CR_01
one_faulty_element_with_multiple_begin_space	TC_03_CR_01
two_or_more_faulty_elements_with_multiple_end_space	TC_04_CR_01
one_faulty_element_with_inner_space	TC_05_CR_01
two_or_more_faulty_elements_with_punctuation	TC_06_CR_01

CR_02 Test Cases

Nome	ID
no_faulty_element	TC_01_CR_02
test_completeness	TC_02_CR_02
test_uniqueness	TC_03_CR_02
test_validity_with_int_value_as_string	TC_04_CR_02
test_validity_with_int_value_as_float	TC_05_CR_02
test_validity_with_float_value_as_string	TC_06_CR_02

CR_03 Test Cases

Nome	ID
test_salaries	TC_01_CR_03
test_red_light	TC_02_CR_03
test_metropolitan_objects	TC_03_CR_03
test_indicators_by_company	TC_04_CR_03
test_food_inspections	TC_05_CR_03
test_aws_cyberattacks	TC_06_CR_03
test_no_smell_dataset	TC_07_CR_03
test_empty_dataset	TC_08_CR_03
test_empty_column_dataset	TC_09_CR_03
test_no_csv_file	TC_10_CR_03

Casi di Test Esistenti

test_detector.py

Modulo che consente di andare a testare il processo di detection di data smell, essa è composta da una classe denominata DetectorTestCase, la quale contiene la lista di risultati attesi, che verificano il corretto funzionamento della detection. Poi vi è la classe di test denominata TestDetectorBuilder, che costruisce il contesto per riuscire a confrontare i risultati attesi, con quelli attuali, che vengono calcolati dal metodo test_creation. Vediamo nel dettaglio i vari test case, raggruppati in un'unica esecuzione.

test_creation

Input

L'input è il dataset denominato "data_smell_testset.csv" che è stato definito proprio per testare la detection corretta dei vari data smell.

Di seguito viene mostrata la composizione di questo dataset.

int1	int2	float1	float2	string1	string2	string3
1	1	1.1	1.1	abc def ghi	abc	abc
2	2	2.2	2.2	abc def ghi	2	-2.2
3	3	3.3	3.8	This is a sentence	-3	3.8
4	4	4.4	4.9	ghi	4	-4.9
5	5	5.5	5.1	cAsing 1	-5	5.1
6	6	6.6	6.2	CaSing 2	6	-6.2
7	7	7.7	7.8	all lowercase	-7	7.8

8	8	8.8	8.9	ALL UPPERCASE	8	-8.9
-300	9	-300.5	9.1	Pseudopseudohypoparathyroidsm	-30	9.1
9	8	9.9	10.2	abc	9	-10.2
10	10	10.2	11.8	def	-10	11.8

Output

Gli output attesi sono più di uno poiché questo test case è stato gestito in modo tale da testare più tipi di smell su un singolo dataset.

Vediamo nel dettaglio tutti gli output attesi nella seguente tabella, dove ogni riga rappresenta un oracolo distinto.

casi di test	column_name	data_smell_type	faulty_elements	statistics
TC1	int1	“Extreme Value Smell”	-300	total_element_count = 11, faulty_element_count = 1
TC2	int1	“Suspect Sign Smell”	-300	total_element_count = 11, faulty_element_count = 1
TC3	float1	“Extreme Value Smell”	-300.5	total_element_count = 11, faulty_element_count = 1
TC4	float1	“Suspect Sign Smell	-300.5	total_element_count = 11, faulty_element_count = 1
TC5	float2	“Integer As Floating Point Number Smell”	1.1, 4.9, 5.1, 8.9, 9.1	total_element_count = 11, faulty_element_count = 5
TC6	string1	“Long Data Value Smell”	"Pseudopseudohypoparathyroidism"	total_element_count = 11, faulty_element_count = 1

TC7	string2	“Integer As String Smell	“2”, “-3”, “4”, “-5”, “6”, “-7”, “8”, “-30”, “9”, “-10”	total_element_count = 11, faulty_element_count = 10
TC8	string3	“Floating Point Number As String Smell”	“-2.2”, “3.8”, “-4.9”, “5.1”, “-6.2”, “7.8”, “-8.9”, “9.1”, “-10.2”, “11.8”	total_element_count = 11, faulty_element_count = 10
TC9	string1	“Casing Smell”	“abc def ghi”, “abc def ghi”, “cAsing 1”, “CaSing 2”, “all lowercase”, “ALL UPPERCASE”	total_element_count = 11, faulty_element_count = 6
TC10	string1	“Duplicated Value Smell”	“abc def ghi”, “abc def ghi”	total_element_count = 11, faulty_element_count = 2
TC11	int2	“Duplicated Value Smell”	8, 8	total_element_count = 11, faulty_element_count = 2

Da notare che, nel test originale l’oracolo era errato poiché non era stato aggiunto il controllo sul total_element_count, infatti negli oracoli originali era 10 ma in realtà è 11. Questo fault è stato individuato grazie ad un’analisi più approfondita del test (infatti la riesecuzione non aveva generato failure).

test_registration.py

Modulo che permette di verificare se l'importazione del modulo datasmelldetection.detectors.great_expectations registri le aspettative corrispondenti per il rilevamento dei data smell (rispetto al registro predefinito).

test_expect_column_values_to_not_contain_missing_value_smell

In questo test è stato utilizzato il metodo check_data_smell_stored_in_registry, che si trova all'interno del modulo helper_functions.py e che consente di verificare se per un ProfilerDataType specifico si può ottenere un dizionario e se nel dizionario è presente quello specifico tipo di data smell, con quel expectation_type. Vediamo nel dettaglio gli input e gli output attesi.

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Missing Value Smell”	“int”, “float, “numeric”, “string”, “boolean”, “datetime”, “unknown	"expect_column_values_to_not_contain_missing_value_smell"

Il registry utilizzato è il default registry che consente di avere un DataSmellRegistry con i valori inizializzati dal suo costruttore.

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

In questo caso l'output atteso è un booleano poiché il controllo effettuato è un semplice assert che verifica se quella condizione è rispettata. L'assert isInstance consente di verificare se smell_dict (il dizionario di smell che vengono prelevati a seconda di uno specifico insieme di ProfilerDataType) è un dizionario. L'assert data_smell_type in smell_dict verifica se il data smell specifico (in questo caso "Missing Value Smell") è presente all'interno del dizionario di data smell. L'assert smell_dict[data_smell_type] == expectation_type confronta se la stringa derivata è uguale a "expect_column_values_to_not_contain_missing_value_smell". Per quanto riguarda i test successivi. i tipi di assert che effettuerà saranno sempre dello stesso tipo poiché viene chiamata la stessa funzionalità, ma con input differenti.

test_expect_column_values_to_not_contain_suspect_sign_smell

Da questo test in poi, la descrizione non varia poiché i tipi di componenti utilizzati sono sempre gli stessi. Vediamo nel dettaglio gli input e gli output attesi.

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Suspect Sign Smell”	“int”, “float, “numeric”	"expect_column_values_to_not_contain_suspect_sign_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expect_column_values_to_not_contain_integer_as_string_smell

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Integer As String Smell”	“string”	"expect_column_values_to_not_contain_integer_as_string_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expect_column_values_to_not_contain_floating_point_number_as_string_smell

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Floating Point Number As String Smell”	“string”	"expect_column_values_to_not_contain_floating_point_number_as_string_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expect_column_values_to_not_contain_extreme_value_smell

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Extreme Value Smell”	“int”, “float”, “numeric”	"expect_column_values_to_not_contain_extreme_value_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expect_column_values_to_not_contain_long_data_value_smell

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Long Data Value Smell”	“string”	"expect_column_values_to_not_contain_long_data_value_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expect_column_values_to_not_contain_integer_as_floating_point_number_smell

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Integer As Floating Point Number Smell”	“float”	"expect_column_values_to_not_contain_integer_as_floating_point_number_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expect_column_values_to_not_contain_duplicated_value_smell

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Duplicated Value Smell”	“int”, “string”	"expect_column_values_to_not_contain_duplicated_value_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expect_column_values_to_not_contain_casing_smell

Input

registry	data_smell_type	profiler_data_types	expectation_type
default_registry	“Casing Smell”	“string”	"expect_column_values_to_not_contain_casing_smell"

Output

assert isInstance(smell_dict, dict)	assert data_smell_type in smell_dict	assert smell_dict[data_smell_type] == expectation_type
True	True	True

test_expectations.py

Modulo che consente di testare se le expectations che implementano il rilevamento dei data smell funzionano come previsto. Gli input dei casi di test sono stati definiti all'interno dei moduli specifici che vengono testati, in un dizionario denominato examples. Ci sarà dunque un dizionario examples per ogni smell, che ne verifica il corretto funzionamento. Inoltre tutti i casi di test sono raggruppati ed eseguiti nello stesso momento all'interno del metodo test_examples_of_all_expectations. Due cose da notare:

- Il test non è stato completato poiché mancano degli examples in tre moduli che sono:
    - expect_column_values_to_not_contain_duplicated_value_smell.py
    - expect_column_values_to_not_contain_extreme_value_smell.py
    - expect_column_values_to_not_contain_missing_value_smell.py
- Il modulo in esame chiama la funzionalità di supporto check_expectation_examples che effettuerà i vari controlli, necessari per garantire il corretto funzionamento.

Di seguito verranno mostrati i vari test case divisi per ogni modulo specifico (dunque per ogni smell).

expect_column_values_to_not_contain_suspect_sign_smell

test_mostly_positive_with_one_negative_number_mostly_1

Input

column	mostly	percentile_threshold
[-1, 0, 1, 2, 3, 4, 5, 6, 8, 9]	1	0.25

Il parametro mostly è un argomento speciale che deve essere compreso tra 0 e 1. Great Expectations lo valuta come percentuale, consentendo un certo margine di manovra nella valutazione delle expectations. Dunque definisce una soglia oltre la quale il risultato viene valutato in maniera positiva o negativa. Il parametro percentile_threshold deve essere compreso nell'intervallo [0,1]. Controlla quali quantili vengono calcolati. I quantili calcolati vengono utilizzati per determinare se la maggior parte dei valori della colonna sono positivi o negativi.

Output

success	partial_unexpected_list
False	[-1]

test_mostly_positive_with_one_negative_number_mostly_0.8

Input

column	mostly	percentile_threshold
[-1, 0, 1, 2, 3, 4, 5, 6, 8, 9]	0.8	0.25

Output

success	partial_unexpected_list
True	[-1]

test_mostly_positive_with_two_negative_numbers_mostly_1

Input

column	mostly	percentile_threshold
[-2, -1, 0, 1, 2, 3, 4, 5, 6, 8]	1	0.25

Output

success	partial_unexpected_list
False	[-1, -2]

test_mostly_positive_with_two_negative_numbers_mostly_0.8

Input

column	mostly	percentile_threshold
[-2, -1, 0, 1, 2, 3, 4, 5, 6, 8]	0.8	0.25

Output

success	partial_unexpected_list
True	[-1, -2]

test_mostly_positive_with_one_positive_number_mostly_1

Input

column	mostly	percentile_threshold
[-8, -7, -6, -5, -4, -3, -2, -1, 0, 1]	1	0.25

Output

success	partial_unexpected_list
False	[1]

test_mostly_positive_with_one_positive_number_mostly_0.8

Input

column	mostly	percentile_threshold
[-8, -7, -6, -5, -4, -3, -2, -1, 0, 1]	0.8	0.25

Output

success	partial_unexpected_list
True	[1]

test_mostly_positive_with_two_positive_numbers_mostly_1

Input

column	mostly	percentile_threshold
[-7, -6, -5, -4, -3, -2, -1, 0, 1, 2]	1	0.25

Output

success	partial_unexpected_list
False	[1, 2]

test_mostly_positive_with_two_positive_numbers_mostly_0.8

Input

column	mostly	percentile_threshold
[-7, -6, -5, -4, -3, -2, -1, 0, 1, 2]	0.8	0.25

Output

success	partial_unexpected_list
True	[1, 2]

test_all_positive

Input

column	mostly	percentile_threshold
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	1	0.25

Output

success	partial_unexpected_list
True	[ ]

test_all_negative

Input

column	mostly	percentile_threshold
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	1	0.25

Output

success	partial_unexpected_list
True	[ ]

expect_column_values_to_not_contain_integer_as_string_smell

test_floats

Input

column
["0.0", "+3.14", "-5.3", "3.", "-5."]

Output

success	partial_unexpected_list
True	[ ]

test_spaces_before

Input

column
[" +1", " -3", " 5", " -4", " 0"]

Output

success	partial_unexpected_list
True	[ ]

test_spaces_after

Input

column
["+1 ", "-3 ", "5 ", "-4 ", "0 "]

Output

success	partial_unexpected_list
True	[ ]

test_integer_in_word

Input

column
["a3b", "abc d-2ef", "a0", "3d", "test abc 3d ef"]

Output

success	partial_unexpected_list
True	[ ]

test_integers_case1_mostly1

Input

column	mostly
["abc", "-2", "5", "", "0"]	1

Output

success	partial_unexpected_list
False	["-2", "5", "0"]

test_integers_case1_mostly1

Input

column	mostly
["abc", "-2", "5", "", "0"]	0.4

Output

success	partial_unexpected_list
True	["-2", "5", "0"]

expect_column_values_to_not_contain_floating_point_number_as_string_smell

test_integers

Input

column
["0", "+3", "-5", "4", "-8"]

Output

success	partial_unexpected_list
True	[ ]

test_spaces_before_short

Input

column
[" 0.", " +3.", " -5.", " 3.", " -8."]

Output

success	partial_unexpected_list
True	[ ]

test_spaces_before_long

Input

column
[" 0.0", " +3.14", " -5.3", " 3.14", " -5.7"]

Output

success	partial_unexpected_list
True	[ ]

test_wrong_point_character_short

Input

column
["0,", "+3,", "-5,", "3,", "-8,"]

Output

success	partial_unexpected_list
True	[ ]

test_wrong_point_character_long

Input

column
["0,0", "+3,14", "-5,3", "3,14", "-5,7"]

Output

success	partial_unexpected_list
True	[ ]

test_spaces_after_short

Input

column
["0. ", "+3. ", "-5. ", "3. ", "-8. "]

Output

success	partial_unexpected_list
True	[ ]

test_spaces_after_long

Input

column
["0.0 ", "+3.14 ", "-5.3 ", "3.14 ", "-5.7 "]

Output

success	partial_unexpected_list
True	[ ]

test_in_word_short

Input

column
["a3.b", "abc d-2.ef", "a0.", "3.d", "test abc 3.d ef"]

Output

success	partial_unexpected_list
True	[ ]

test_in_word_long

Input

column
["a3.14b", "abc d-2.84ef", "a0.0", "3.3d", "test abc 3.14d ef"]

Output

success	partial_unexpected_list
True	[ ]

test_floats_short

Input

column	mostly
["0.", "+3.", "-5.", "3.", "-8."]	1

Output

success	partial_unexpected_list
False	"0.", "+3.", "-5.", "3.", "-8."]

test_floats_long

Input

column	mostly
["0.0", "+3.14", "-5.3", "3.14", "-5.7"]	1

Output

success	partial_unexpected_list
False	["0.0", "+3.14", "-5.3", "3.14", "-5.7"]

test_floats_case1_mostly1

Input

column	mostly
["abc", "-2.", "5.8", "", "0.0"]	1

Output

success	partial_unexpected_list
False	["-2.", "5.8", "0.0"]

test_floats_case1_mostly0.4

Input

column	mostly
["abc", "-2.", "5.8", "", "0.0"]	1

Output

success	partial_unexpected_list
True	["-2.", "5.8", "0.0"]

expect_column_values_to_not_contain_long_data_value_smell

test_no_spaces_threshold_50

Input

column	length_threshold
["word", "Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "", "Pseudopseudohypoparathyroidism"]	50

Output

success	partial_unexpected_list
True	[ ]

test_no_spaces_threshold_30_1_mostly

Input

column	length_threshold	mostly
["word", "Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "", "Pseudopseudohypoparathyroidism"]	30	1

Output

success	partial_unexpected_list
False	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_no_spaces_threshold_30_0.4_mostly

Input

column	length_threshold	mostly
["word", "Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "", "Pseudopseudohypoparathyroidism"]	30	0.4

Output

success	partial_unexpected_list
True	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_no_spaces_threshold_20_1_mostly

Input

column	length_threshold	mostly
["word", "Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "", "Pseudopseudohypoparathyroidism"]	20	1

Output

success	partial_unexpected_list
False	["Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_spaces_before_threshold_50

Input

column	length_threshold
[" word", " Pneumonoultramicroscopicsilicovolcanoconiosis", " ", " Incomprehensibilities", " Pseudopseudohypoparathyroidism"]	50

Output

success	partial_unexpected_list
True	[ ]

test_spaces_before_threshold_30_1_mostly

Input

column	length_threshold	mostly
[" word", " Pneumonoultramicroscopicsilicovolcanoconiosis", " ", " Incomprehensibilities", " Pseudopseudohypoparathyroidism"]	30	1

Output

success	partial_unexpected_list
False	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_spaces_before_threshold_30_0.4_mostly

Input

column	length_threshold	mostly
[" word", " Pneumonoultramicroscopicsilicovolcanoconiosis", " ", " Incomprehensibilities", " Pseudopseudohypoparathyroidism"]	30	0.4

Output

success	partial_unexpected_list
True	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_spaces_before_threshold_20_1_mostly

Input

column	length_threshold	mostly
[" word", " Pneumonoultramicroscopicsilicovolcanoconiosis", " ", " Incomprehensibilities", " Pseudopseudohypoparathyroidism"]	20	1

Output

success	partial_unexpected_list
False	["Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_spaces_after_threshold_50

Input

column	length_threshold
["Pseudopseudohypoparathyroidism ", "word ", " ", "Incomprehensibilities ", "Pneumonoultramicroscopicsilicovolcanoconiosis "]	50

Output

success	partial_unexpected_list
True	[ ]

test_spaces_after_threshold_30_1_mostly

Input

column	length_threshold	mostly
["Pseudopseudohypoparathyroidism ", "word ", " ", "Incomprehensibilities ", "Pneumonoultramicroscopicsilicovolcanoconiosis "]	30	1

Output

success	partial_unexpected_list
False	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_spaces_after_threshold_30_0.4_mostly

Input

column	length_threshold	mostly
["Pseudopseudohypoparathyroidism ", "word ", " ", "Incomprehensibilities ", "Pneumonoultramicroscopicsilicovolcanoconiosis "]	30	0.4

Output

success	partial_unexpected_list
True	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_spaces_after_threshold_20_1_mostly

Input

column	length_threshold	mostly
["Pseudopseudohypoparathyroidism ", "word ", " ", "Incomprehensibilities ", "Pneumonoultramicroscopicsilicovolcanoconiosis "]	20	1

Output

success	partial_unexpected_list
False	["Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_before_threshold_50

Input

column	length_threshold
["A test word", "Unrelated words Pneumonoultramicroscopicsilicovolcanoconiosis", "Testcase ", "Another unrelated word Incomprehensibilities", "Test Pseudopseudohypoparathyroidism"]	50

Output

success	partial_unexpected_list
True	[ ]

test_words_before_threshold_30_1_mostly

Input

column	length_threshold	mostly
["A test word", "Unrelated words Pneumonoultramicroscopicsilicovolcanoconiosis", "Testcase ", "Another unrelated word Incomprehensibilities", "Test Pseudopseudohypoparathyroidism"]	30	1

Output

success	partial_unexpected_list
False	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_before_threshold_30_0.4_mostly

Input

column	length_threshold	mostly
["A test word", "Unrelated words Pneumonoultramicroscopicsilicovolcanoconiosis", "Testcase ", "Another unrelated word Incomprehensibilities", "Test Pseudopseudohypoparathyroidism"]	30	0.4

Output

success	partial_unexpected_list
True	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_before_threshold_20_1_mostly

Input

column	length_threshold	mostly
["A test word", "Unrelated words Pneumonoultramicroscopicsilicovolcanoconiosis", "Testcase ", "Another unrelated word Incomprehensibilities", "Test Pseudopseudohypoparathyroidism"]	20	1

Output

success	partial_unexpected_list
False	["Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_after_threshold_50

Input

column	length_threshold
["Pseudopseudohypoparathyroidism unrelated words", "word test", " short word", "Incomprehensibilities unrelated part", "Pneumonoultramicroscopicsilicovolcanoconiosis testcase"]	50

Output

success	partial_unexpected_list
True	[ ]

test_words_after_threshold_30_1_mostly

Input

column	length_threshold	mostly
["Pseudopseudohypoparathyroidism unrelated words", "word test", " short word", "Incomprehensibilities unrelated part", "Pneumonoultramicroscopicsilicovolcanoconiosis testcase"]	30	1

Output

success	partial_unexpected_list
False	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_after_threshold_30_0.4_mostly

Input

column	length_threshold	mostly
["Pseudopseudohypoparathyroidism unrelated words", "word test", " short word", "Incomprehensibilities unrelated part", "Pneumonoultramicroscopicsilicovolcanoconiosis testcase"]	30	0.4

Output

success	partial_unexpected_list
True	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_after_threshold_20_1_mostly

Input

column	length_threshold	mostly
["Pseudopseudohypoparathyroidism unrelated words", "word test", " short word", "Incomprehensibilities unrelated part", "Pneumonoultramicroscopicsilicovolcanoconiosis testcase"]	20	1

Output

success	partial_unexpected_list
False	["Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_before_and_after_threshold_50

Input

column	length_threshold
["A test word Pseudopseudohypoparathyroidism unrelated words", "Unrelated words word test", "Another unrelated word Incomprehensibilities unrelated part", "Test Pneumonoultramicroscopicsilicovolcanoconiosis testcase", "Testcase short word",]	50

Output

success	partial_unexpected_list
True	[ ]

test_words_before_and_after_threshold_30_1_mostly

Input

column	length_threshold	mostly
["A test word Pseudopseudohypoparathyroidism unrelated words", "Unrelated words word test", "Another unrelated word Incomprehensibilities unrelated part", "Test Pneumonoultramicroscopicsilicovolcanoconiosis testcase", "Testcase short word",]	30	1

Output

success	partial_unexpected_list
False	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_before_and_after_threshold_30_0.4_mostly

Input

column	length_threshold	mostly
["A test word Pseudopseudohypoparathyroidism unrelated words", "Unrelated words word test", "Another unrelated word Incomprehensibilities unrelated part", "Test Pneumonoultramicroscopicsilicovolcanoconiosis testcase", "Testcase short word",]	30	0.4

Output

success	partial_unexpected_list
True	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_words_before_and_after_threshold_20_1_mostly

Input

column	length_threshold	mostly
["A test word Pseudopseudohypoparathyroidism unrelated words", "Unrelated words word test", "Another unrelated word Incomprehensibilities unrelated part", "Test Pneumonoultramicroscopicsilicovolcanoconiosis testcase", "Testcase short word",]	20	1

Output

success	partial_unexpected_list
False	["Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

test_no_spaces_default

Input

column
["word", "Incomprehensibilities", "Pneumonoultramicroscopicsilicovolcanoconiosis", "", "Pseudopseudohypoparathyroidism"]

Output

success	partial_unexpected_list
False	["Pneumonoultramicroscopicsilicovolcanoconiosis", "Pseudopseudohypoparathyroidism"]

expect_column_values_to_not_contain_integer_as_floating_point_number_smell

test_distance0.1and0.2_eps_0.5_mostly_1

Input

column	mostly	epsilon
[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]	1	0.5

Il parametro epsilon rappresenta la soglia per la differenza assoluta di un numero in virgola mobile, al numero intero più vicino. Questo parametro deve essere un valore in virgola mobile maggiore di zero.

Output

success	partial_unexpected_list
False	[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]

test_distance0.1and0.2_eps_0.25_mostly_1

Input

column	mostly	epsilon
[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]	1	0.25

Output

success	partial_unexpected_list
False	[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]

test_distance0.1and0.2_eps_0.15_mostly_1

Input

column	mostly	epsilon
[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]	1	0.15

Output

success	partial_unexpected_list
False	[-5.9, -0.1, 0.1, 7.9, 8.1]

test_distance0.1and0.2_eps_0.15_mostly_0.4

Input

column	mostly	epsilon
[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]	0.4	0.15

Output

success	partial_unexpected_list
True	[-5.9, -0.1, 0.1, 7.9, 8.1]

test_distance0.1and0.2_eps_0.15_mostly_0.5

Input

column	mostly	epsilon
[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]	0.5	0.15

Output

success	partial_unexpected_list
True	[-5.9, -0.1, 0.1, 7.9, 8.1]

test_distance0.1and0.2_eps_0.05_mostly_1

Input

column	mostly	epsilon
[-5.8, -5.9, -0.2, -0.1, 0.1, 0.2, 7.8, 7.9, 8.1, 8.2]	1	0.05

Output

success	partial_unexpected_list
True	[ ]

expect_column_values_to_not_contain_casing_smell

test_lowercase_only_wordcount1_mostly1

Input

column	same_case_wordcount_threshold	mostly
["abc", "abc def", "abc def ghi", "abc def ghi jkl", ""]	1	1

Il parametro same_case_wordcount_threshold controlla quante parole devono essere contenute in una stringa perché esista un casing smell, in cui tutte le parole hanno lo stesso caso. Per esempio, stringhe come "tutto minuscolo" o "TUTTO MAIUSCOLO" con `soglia di numero di parole con lo stesso caso = 2' attiveranno questo caso, mentre "stringa" non lo farà.

Output

success	partial_unexpected_list
False	["abc", "abc def", "abc def ghi", "abc def ghi jkl"]

test_lowercase_only_wordcount2_mostly1

Input

column	same_case_wordcount_threshold	mostly
["abc", "abc def", "abc def ghi", "abc def ghi jkl", ""]	2	1

Output

success	partial_unexpected_list
False	["abc def", "abc def ghi", "abc def ghi jkl"]

test_lowercase_only_wordcount3_mostly1

Input

column	same_case_wordcount_threshold	mostly
["abc", "abc def", "abc def ghi", "abc def ghi jkl", ""]	3	1

Output

success	partial_unexpected_list
False	["abc def ghi", "abc def ghi jkl"]

test_lowercase_only_wordcount3_mostly0.6

Input

column	same_case_wordcount_threshold	mostly
["abc", "abc def", "abc def ghi", "abc def ghi jkl", ""]	3	0.6

Output

success	partial_unexpected_list
True	["abc def ghi", "abc def ghi jkl"]

test_uppercase_only_wordcount1_mostly1

Input

column	same_case_wordcount_threshold	mostly
["ABC", "ABC DEF", "ABC DEF GHI", "ABC DEF GHI JKL", "Unrelated"]	1	1

Output

success	partial_unexpected_list
False	["ABC", "ABC DEF", "ABC DEF GHI", "ABC DEF GHI JKL"]

test_uppercase_only_wordcount2_mostly1

Input

column	same_case_wordcount_threshold	mostly
["ABC", "ABC DEF", "ABC DEF GHI", "ABC DEF GHI JKL", "Unrelated"]	2	1

Output

success	partial_unexpected_list
False	["ABC DEF", "ABC DEF GHI", "ABC DEF GHI JKL"]

test_uppercase_only_wordcount3_mostly1

Input

column	same_case_wordcount_threshold	mostly
["ABC", "ABC DEF", "ABC DEF GHI", "ABC DEF GHI JKL", "Unrelated"]	3	1

Output

success	partial_unexpected_list
False	["ABC DEF GHI", "ABC DEF GHI JKL"]

test_uppercase_only_wordcount3_mostly0.6

Input

column	same_case_wordcount_threshold	mostly
["ABC", "ABC DEF", "ABC DEF GHI", "ABC DEF GHI JKL", "Unrelated"]	3	0.6

Output

success	partial_unexpected_list
True	["ABC DEF GHI", "ABC DEF GHI JKL"]

test_mixed_case_only_mostly1

Input

column	same_case_wordcount_threshold	mostly
["Abc dE", "AbC", "AbcDe", "abc dEf ghi", "Unrelated"]	3	1

Output

success	partial_unexpected_list
False	["Abc dE", "AbC", "AbcDe", "abc dEf ghi"]

test_mixed_case_only_mostly0.2

Input

column	same_case_wordcount_threshold	mostly
["Abc dE", "AbC", "AbcDe", "abc dEf ghi", "Unrelated"]	3	0.2

Output

success	partial_unexpected_list
True	["Abc dE", "AbC", "AbcDe", "abc dEf ghi"]

test_sentences_wordcount2_mostly1

Input

column	same_case_wordcount_threshold	mostly
["This is an example sentence.", "This is an eXample sentence.", "This is an ExamPle sentence.", "this is an example sentence.", "THIS IS AN EXAMPLE SENTENCE."]	2	1

Output

success	partial_unexpected_list
False	["This is an eXample sentence.", "This is an ExamPle sentence.", "this is an example sentence.", "THIS IS AN EXAMPLE SENTENCE."]

test_sentences_wordcount5_mostly1

Input

column	same_case_wordcount_threshold	mostly
["This is an example sentence.", "This is an eXample sentence.", "This is an ExamPle sentence.", "this is an example sentence.", "THIS IS AN EXAMPLE SENTENCE."]	5	1

Output

success	partial_unexpected_list
False	["This is an eXample sentence.", "This is an ExamPle sentence.", "this is an example sentence.", "THIS IS AN EXAMPLE SENTENCE."]

test_sentences_wordcount6_mostly1

Input

column	same_case_wordcount_threshold	mostly
["This is an example sentence.", "This is an eXample sentence.", "This is an ExamPle sentence.", "this is an example sentence.", "THIS IS AN EXAMPLE SENTENCE."]	6	1

Output

success	partial_unexpected_list
False	["This is an eXample sentence.", "This is an ExamPle sentence."]

test_sentences_wordcount6_mostly0.6

Input

column	same_case_wordcount_threshold	mostly
["This is an example sentence.", "This is an eXample sentence.", "This is an ExamPle sentence.", "this is an example sentence.", "THIS IS AN EXAMPLE SENTENCE."]	6	0.6

Output

success	partial_unexpected_list
True	["This is an eXample sentence.", "This is an ExamPle sentence."]

test_negative_wordcount1_mostly1

Input

column	same_case_wordcount_threshold	mostly
["-1", "5", "2.5", "-3.8", ""]	1	1

Output

success	partial_unexpected_list
True	[ ]

test_negative_wordcount2_mostly1

Input

column	same_case_wordcount_threshold	mostly
["A test sentence which should not be flagged.", "www.google.de", "WWW.GOOGLE.DE", "www.gOogle.de", "www.GooGle.de"]	2	1

Output

success	partial_unexpected_list
True	[ ]