1. Introduzione e panoramica del sistema

CodeSmile è uno strumento di analisi statica progettato per rilevare code smells specifici per il Machine Learning (ML-CSs) e riportarli tramite file .csv con informazioni come file name, function name, line e nome dello smell individuato. Questo documento si propone di progettare la fase di testing del sistema software, creando tests sia per le funzionalità già esistenti, modificate e nuove introdotte dalle change requests.

1.1 Test suite attuale

Il software esistente non presenta alcuna documentazione o file di testing, ad eccezione di test_models.py. Quest'ultimo, in maniera white-box, testa il file models.py che si occupa di identificare i modelli più comuni di machine learning.

2. Features da testare

Di seguito la lista delle features di cui si effettuerà il testing, in **grassetto** quelle di cui si effettuerà test di regressione:

- **Analisi code smells progetto (singola, sequenziale e parallela) (UC_1)**
- Generazione reports numero di code smells per progetto e numero di code smells per tipo (UC_2)
- Generazione grafico a torta o a barre (UC_3)
- Generazione grafico temporale (UC_4)

3. Pass/fail criteria

Le attività di testing sono mirate ad identificare la presenza di faults all'interno del sistema, per effettuarne un successivo intervento di eliminazione.
L'esito di un test case è valutato mediante un oracolo, inteso come il risultato atteso della sua esecuzione, basandosi sui requisiti.
Un test (insieme di test cases) ha successo (pass) se, per almeno un test case, l'output del sistema è diverso dall'output dell'oracolo, rivelando quindi un fault.
Un test (insieme di test cases) fallisce (fail) se, per tutti i test cases, l'output del sistema è uguale all'output atteso dall'oracolo, non rivelando quindi nessun fault.

4. Approccio

4.1 Functional Testing

Verrà eseguito tramite category partition, basandosi e testando tutti gli use cases del sistema.

4.2 Unit Testing

Verrà eseguito per le seguenti rules che individuano code smells:

1. Deterministic Algorithm Option Not Used
2. Hyperparameters not explicitly set (CR_1)
3. Randomness uncontrolled (CR_1)

Inoltre verrà eseguito per le funzioni di utilità: *dataframe_check* e *recursive_search_variables* contenute in *dataframe_detector.py*

4.3 Integration Testing

Verrà fatto un integration testing bottom-up della rule *merge_api_parameter_not_explictly_set*, testando le funzioni da essa chiamate:

- *dataframe_check*
- *dataframe_detector.py*

che come menzionato prima verranno testate a livello di unità

5. Materiale di testing

TSL generator (https://github.com/alexorso/tslgenerator) è stato impiegato per generare i test frames del category partition di Analisi code smells progetto. La libreria unittest di Python verrà usata per i test di unità

6. Category partition

6.1 Analisi code smells progetto

Parametro: input	
Nome Categoria	**Scelta per la categoria**
Contenuto Input [CI]	1. Contiene una cartella con un singolo progetto python **[property SINGLEPROJ]** 2. Contiene una cartella con molteplici progetti python (2 <= N <= 10) **[property MULTIPROJ]** 3. Contiene una cartella con molteplici progetti python (N >= 11) **[property MULTIPROJ]** 4. Non contiene files python **[error]** 5. Il path non esiste **[error]** 6. Contiene solo files di test python **[error]** 7. Il path esiste ma non ci sono files **[error]** 8. Il path non è una cartella **[error]**
Size progetto [SP]	1. 1 <= files python <= 10 **[if SINGLEPROJ]** 2. 11 <= files python <= 50 **[if SINGLEPROJ]** 3. files python >= 51 **[if SINGLEPROJ]**
Parametro: output	
Nome Categoria	**Scelta per la categoria**
Esistenza [E]	1. Contiene una cartella esistente e scrivibile 2. Contiene una cartella esistente ma non scrivibile **[error]** 3. Contiene una cartella non esistente

	[error] 4. Contiene un file **[error]**
Parametro: multiple	
Nome Categoria	**Scelta per la categoria**
Flag multiple [Fm]	1. True **[property MULTIPLE]** 2. False
Parametro: parallel	
Nome Categoria	**Scelta per la categoria**
Flag parallel [Fp]	1. True **[if MULTIPLE] [property PARALLEL]** 2. False
Parametro: resume	
Nome Categoria	**Scelta per la categoria**
Flag resume [Fr]	1. True **[if MULTIPLE && !PARALLEL] [property RESUME]** 2. False
Esistenza Execution Log [EEL]	1. L'execution_log esiste **[if RESUME]** 2. L'execution_log non esiste **[if RESUME] [error]**
Parametro: max_workers	
Nome Categoria	**Scelta per la categoria**
Numero Threads [NT]	1. Non selezionato 2. 0 **[if PARALLEL] [error]** 3. 1 <= NT <= 10 **[if PARALLEL]** 4. > 10 **[if PARALLEL]**

Test cases qui:
https://github.com/SugarStoneMaster/smell_ai/blob/main/test/analyzer/testcases

6.2 Generazione reports ausiliari

Parametro: input	
Nome Categoria	**Scelta per la categoria**
Esistenza [E]	1. Contiene il file overview_output.csv 2. Non contiene il file overview_output.csv **[error]**
Formato File [FF]	1. Contiene le colonne “filename”, “name_smell” e “smell” 2. Contiene le colonne “filename”, “name_smell” e “smell” ma non sono di tipo (stringa, stringa, intero) **[error]** 3. Manca almeno una colonna tra “filename”, “name_smell” e “smell” **[error]** 4. Il file è vuoto **[error]** 5. Il file non è in formato .csv **[error]**

Test Case ID	**Test Frame**	**Esito**
TC_2.1	**E1, FF1**	Vengono generati i reports ausiliari
TC_2.2	**E1, FF2**	**Invalid**: le colonne non sono del tipo giusto
TC_2.3	**E1, FF3**	**Invalid**: mancano le colonne necessarie
TC_2.4	**E1, FF4**	**Invalid**: il file è vuoto

TC_2.5	**E1, FF5**	**Invalid**: il file non è in formato .csv
TC_2.6	**E2, FF1**	**Invalid**: il file non esiste

6.3 Generazione grafico a torta o a barre

Parametro: input	
Nome Categoria	**Scelta per la categoria**
Esistenza [E]	1. Contiene il file general_overview.csv 2. Non contiene il file general_overview.csv **[error]**
Formato File [FF]	1. Contiene le colonne "name_smell" e "smell" 2. Contiene le colonne "name_smell" e "smell" ma non sono di tipo (stringa, intero) **[error]** 3. Manca almeno una colonna tra "name_smell" e "smell" **[error]** 4. Il file è vuoto **[error]** 5. Il file non è in formato .csv **[error]**
Parametro: pie	
Nome Categoria	**Scelta per la categoria**
Flag [F]	1. True 2. False

Test Case ID	**Test Frame**	**Esito**

TC_3.1	**E1, FF1, F1**	Viene generato il grafico a torta correttamente
TC_3.2	**E1, FF1, F2**	Viene generato il grafico a barre
TC_3.3	**E1, FF1, F3**	**Invalid**: valore flag non valido
TC_3.4	**E1, FF2, F1**	**Invalid**: le colonne non sono del tipo giusto
TC_3.5	**E1, FF3, F1**	**Invalid**: mancano le colonne necessarie
TC_3.6	**E1, FF4, F1**	**Invalid**: il file è vuoto
TC_3.7	**E1, FF5, F1**	**Invalid**: il file non è in formato .csv
TC_3.8	**E2, FF1, F1**	**Invalid**: il file non esiste

6.4 Generazione grafico temporale

Parametro: input	
Nome Categoria	**Scelta per la categoria**
Esistenza [E]	1. Contiene il file smell_count_dates.csv 2. Non contiene il file smell_count_dates.csv **[error]**

Formato File [FF]	1. Contiene le colonne “smells” e “date” 2. Contiene le colonne “smells” e “date” ma non sono di tipo (intero, datetime) **[error]** 3. Manca almeno una colonna tra “smells” e “date” **[error]** 4. Il file è vuoto **[error]** 5. Il file non è in formato .csv **[error]**

Test Case ID	Test Frame	Esito
TC_4.1	**E1, FF1**	Viene generato il grafico temporale correttamente
TC_4.2	**E1, FF2**	**Invalid**: le colonne non sono del tipo giusto
TC_4.3	**E1, FF3**	**Invalid**: mancano le colonne necessarie
TC_4.4	**E1, FF4**	**Invalid**: il file è vuoto
TC_4.5	**E1, FF5**	**Invalid**: il file non è in formato .csv
TC_4.6	**E2, FF1**	**Invalid**: il file non esiste