UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Ingegneria, Gestione ed Evoluzione del Software

Pre-Modification Integration Test Plan

TEAM MEMBER

Donia Daniele - 0522501575

La Marca Antonio - 0522501557

Somma Pasquale - 0522501543

Indice

CAPITOLO 1

Introduzione

1.1 Approccio

Considerando che il tool prevede l'esecuzione individuale di ciascuno script, abbiamo deciso di testare insieme gli script all'interno di ogni singolo modulo. Questa scelta è motivata dal fatto che, all'interno di un modulo, l'output di uno script spesso rappresenta l'input per lo script successivo. Inoltre, nonostante gli script siano separati, essi vengono considerati esternamente come un unico blocco, poiché solo l'output finale è utilizzato dagli altri moduli, mentre gli output intermedi non vengono presi in considerazione.

CAPITOLO 2

Test di integrazione

2.1 Componenti testate

2.1.1 Repo Mining

Gli script presenti in questo modulo vengono eseguiti nel seguente ordine:

1. `divide_dataset.py`
2. `main_repo_Mining.py`

Per testare gli script e la loro interazione abbiamo individuato i seguenti oggetti dell'ambiente e le relative categorie:

- **Oggetto dell'ambiente 1:** initial_dataset.csv
  - Categoria: Esistenza (EI)
    - Scelta 1: Esiste (EI1) [property dataExistOk]
    - Scelta 2: Non esiste (EI2) [error]
  - Categoria: Formato (FI)
    - Scelta 1: Formato valido (FD1) [if dataExistOk] [property dataFormatOk]
    - Scelta 2: Formato non valido (FD2) [error]
  - Categoria: Numero record (N)
    - Scelta 1: 0 (N1) [if dataExistOk]

    - Scelta 2: $\leq$50 (N2) [if dataExistOk] [property dataNotEmpty]
    - Scelta 3: >50 (N3) [if dataExistOk] [property dataNotEmpty] [single]
  - Categoria: Contenuto link (CL)
    - Scelta 1: Link esistente e valido (CL1) [if dataFormatOk and dataNotEmpty] [propery linkValidOk]
    - Scelta 2: Link valido e non esistente (CL2) [if dataFormatOk and dataNotEmpty]
    - Scelta 3: Link non valido (CL3) [if dataFormatOk and dataNotEmpty]
  - Categoria: Contenuto repo (CR)
    - Scelta 1: Presente repo valida (CR1) [if linkValidOk] [property repoValidOk]
    - Scelta 2: Presente repo non valida (CR2) [if linkValidOk]
  - Categoria: Contenuto commit (CC)
    - Scelta 1: Presente commit valido (CC1) [if repoValidOk] [property commitValidOk]
    - Scelta 2: Presente commit valido con path che eccede la lunghezza massima del S.O. (CC2) [if repoValidOk]
    - Scelta 3: Presente commit non valido con risposta non riuscita (CC3) [if repoValidOk]
    - Scelta 4: Presente commit non valido con risposta riuscita (CC4) [if repoValidOk]
  - Categoria: Contenuto modifica (CM)
    - Scelta 1: Non esiste una modifica per un commit (CM1) [if commitValidOk]
    - Scelta 2: Esiste una modifica per commit e non comprende classi java (CM2) [if commitValidOk]
    - Scelta 3: Esiste una modifica per commit che riguarda classi .java introdotte con esso (CM3) [if commitValidOk]
    - Scelta 4: Esiste una modifica per commit che riguarda classi .java già presenti prima di esso (CM4) [if commitValidOk]

- **Oggetto dell'ambiente 2:** directory Dataset_Divided
  - Categoria: Esistenza (ED)
    - Scelta 1: Esiste (ED1)

        * Scelta 2: Non esiste (ED2) [single]

- **Oggetto dell'ambiente 3:** directory mining_results
  - Categoria: Esistenza (EM)
    * Scelta 1: Esiste (EM1)
    * Scelta 2: Non esiste (EM2) [error]
- **Oggetto dell'ambiente 4:** file CHECK.txt
  - Categoria: Esistenza (EC)
    * Scelta 1: Esiste (EC1)
    * Scelta 2: Non esiste (EC2) [single]
- **Oggetto dell'ambiente 5:** file ERRORS.txt
  - Categoria: Esistenza (EE)
    * Scelta 1: Esiste (EE1)
    * Scelta 2: Non esiste (EE2) [single]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_1.1	EI2	FileNotFoundError
TCI_1.2	EM2	FileNotFoundError
TCI_1.3	EI1 FD2	KeyError
TCI_1.4	EI1 FI1 N1 ED2 EM1 EC2 EE2	Dataset_Divided creata e contiene 1.csv e mining_results vuota
TCI_1.5	EI1 FI1 N2 CL3	Missing Schema Error
TCI_1.6	EI1 FI1 N2 CL2	Connection Error
TCI_1.7	EI1 FI1 N3 CL1 CR2 ED1 EM1 EC2 EE2	CHECKS.txt indica che la repo non è disponibile
TCI_1.8	EI1 FI1 N2 CL1 CR1 CC3 ED1 EM1 EC2 EE2	CHECKS.txt indica che il commit non è esistente
TCI_1.9	EI1 FI1 N2 CL1 CR1 CC4 ED1 EM1 EC1 EE1	CHECKS.txt e ERRORS.txt indicano che il commit non è definito

Test Case	Combinazione	Oracolo
TCI_1.10	EI1 FI1 N2 CL1 CR1 CC2 ED1 EM1 EC2 EE2	GitCommandError
TCI_1.11	EI1 FI1 N2 CL1 CR1 CC1 CM1 ED1 EM1 EC2 EE2	CHECKS.txt indica status ok
TCI_1.12	EI1 FI1 N2 CL1 CR1 CC1 CM2 ED1 EM1 EC2 EE2	CHECKS.txt indica status ok
TCI_1.13	EI1 FI1 N2 CL1 CR1 CC1 CM3 ED1 EM1 EC2 EE2	CHECKS.txt indica status ok e dir. con id commit creata e vuota
TCI_1.14	EI1 FI1 N2 CL1 CR1 CC1 CM4 ED1 EM1 EC2 EE2	CHECKS.txt indica status ok e dir. con id commit contiene java file modificati e già esistenti

Tabella 2.1: Test case per il modulo Repo Mining

2.1.2 Text Mining

Gli script presenti in questo modulo vengono eseguiti nel seguente ordine:

1. `text_mining.py`
2. `dict_generator.py`
3. `less_element_text_mining.py`
4. `creator_csv_for_TextMining.py`

Per testare gli script e la loro interazione abbiamo individuato le seguenti categorie di parametri, operando sulla presenza e assenza delle directory, dei file e dei loro contenuti:

- **Oggetto dell'ambiente 1**: mining_results
  - Categoria: Directory e file (DF)
    - Scelta 1: Directory non esistente (DF1) [error]
    - Scelta 2: Directory esistente (DF2) [property miningOk]
- **Parametro 1**: repo
  - Categoria: Esistenza directory (ES)

        * Scelta 1: Directory non presente (ES1) [error]
        * Scelta 2: Directory presente (ES2) [if miningOk property repoOk]
    - Categoria: Contenuto directory (CN)
        * Scelta 1: Directory vuota (CN1) [error]
        * Scelta 2: Directory non vuota (CN2) [if repoOk property repoNotEmpty]

- **Parametro 2**: cvd_id
    - Categoria: Stato Parametro (STC)
        * Scelta 1: Il parametro è un file (STC1) [error]
        * Scelta 2: Il parametro è un file non ammesso (".DS_Store", "CHECK.txt", "ERRORS.txt") (STC2) [error]
        * Scelta 3: Il parametro è una cartella (STC3) [if repoNotEmpty property cvdOk]
    - Categoria: Contenuto directory (CNC)
        * Scelta 1: La directory è vuota (CNC1) [error]
        * Scelta 2: La directory non è vuota (CNC2) [if cvdOk property cvdNotEmpty]

- **Parametro 3**: folder
    - Categoria: Stato Parametro (STF)
        * Scelta 1: il parametro è un file (STF1) [error]
        * Scelta 2: il parametro è un file non ammesso (STF2) [error]
        * Scelta 3: il parametro è una cartella (STF3) [if cvdNotEmpty property folderOk]
    - Categoria: Contenuto directory (CNF)
        * Scelta 1: La directory è vuota (CNF1) [error]
        * Scelta 2: La directory non è vuota (CNF2) [if folderOk property folderNotEmpty]

- **Parametro 4**: file
    - Categoria: Contenuto directory (CNFI)
        * Scelta 1: il parametro è file non ammesso (CNFI1) [if folderNotEmpty]
        * Scelta 2: il parametro è una directory (CNFI2) [error]
        * Scelta 3: il parametro è un file Java (CNFI3) [if folderNotEmpty property fileOk]

        * Scelta 4: il parametro è un file non Java (CNFI4) [if folderNotEmpty property fileOk] [single]
    - Categoria: Contenuto file (CFJ)
        * Scelta 1: Il file è vuoto (CFJ1) [if fileOk]
        * Scelta 2: Il file non è vuoto (CFJ2) [if fileOk]
    - Categoria: Esistenza file (ESTM)
        * Scelta 1: File di text_mining.txt già esistente (ESTM1) [if folderNotEmpty property existOk]
        * Scelta 2: File di text_mining.txt non esistente (ESTM2) [if folderNotEmpty]
    - Categoria: Contenuto file di text mining (CFTM)
        * Scelta 1: Il file è vuoto (CFTM1) [if existOk]
        * Scelta 2: Il file non è vuoto (CFTM2) [if existOk]

- **Parametro 5**: text_mining_dict
    - Categoria: Esistenza file (ESD)
        * Scelta 1: File già esistente e scritto (ESD1) [if miningOk] [single]
        * Scelta 2: File non esistente (ESD2) [if miningOk]

- **Parametro 6**: Filtered_text_mining
    - Categoria: Esistenza file (ESF)
        * Scelta 1: File già esistente e scritto (ESF1) [if miningOk] [single]
        * Scelta 2: File non esistente (ESF2) [if miningOk]

- **Parametro 7**: csv_mining_final
    - Categoria: Esistenza file (ESC)
        * Scelta 1: File già esistente e scritto (ESC1) [if miningOk] [single]
        * Scelta 2: File non esistente (ESC2) [if fminingOk]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_2.1	DF1	FileNotFoundError

Test Case	Combinazione	Oracolo
TCI_2.2	DF2 ES1	FileNotFoundError
TCI_2.3	DF2 ES2 CN1 ESD2 ESF2 ESC2	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.4	DF2 ES2 CN2 STC1 ESD2 ESF2 ESC2	NotADirectoryError
TCI_2.5	DF2 ES2 CN2 STC2 ESD2 ESF2 ESC2	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.6	DF2 ES2 CN2 STC3 CNC1 ESD2 ESF2 ESC2	Il file text_mining.txt non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.7	DF2 ES2 CN2 STC3 CNC2 STF1 ESD2 ESF2 ESC2	NotADirectoryError
TCI_2.8	DF2 ES2 CN2 STC3 CNC2 STF2 ESD2 ESF2 ESC2	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.9	DF2 ES2 CN2 STC3 CNC2 STF3 CNF1 ESD2 ESF2 ESC2	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.10	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 ESD2 ESF2 ESC2	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.11	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI2 ESD2 ESF2 ESC2	PermissionError
TCI_2.12	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ1 ESTM2 ESD2 ESF2 ESC2	Il file text_mining.txt viene creato vuoto e gli altri .txt vengono creati con dizionari vuoti e il csv solo con l'intestazione
TCI_2.13	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ1 ESTM1 CFTM1 ESD2 ESF2 ESC2	Il file text_mining.txt viene sovrascritto e gli altri .txt vengono creati con dizionari vuoti e il csv solo con l'intestazione
TCI_2.14	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ1	Il file text_mining.txt viene sovrascritto e gli altri .txt vengono creati con

Test Case	Combinazione	Oracolo
	ESTM1 CFTM2 ESD2 ESF2 ESC2	dizionari vuoti e il csv solo con l'intestazione
TCI_2.15	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ2 ESTM2 ESD2 ESF2 ESC2	Il file text_mining.txt viene creato, col dizionario relativo, e anche gli altri file vengono creati con dizionari non vuoti, e il csv con una riga per ogni file
TCI_2.16	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ2 ESTM2 ESD1 ESF1 ESC1	Il file text_mining.txt viene creato, col dizionario relativo, e gli altri file vengono sovrascritti con dizionari non vuoti, e il csv con una riga per ogni file
TCI_2.17	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ2 ESTM1 CFTM1 ESD2 ESF2 ESC2	Il file text_mining.txt viene sovrascritto, col dizionario relativo, e anche gli altri file vengono creati con dizionari non vuoti, e il csv con una riga per ogni file
TCI_2.18	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ2 ESTM1 CFTM2 ESD2 ESF2 ESC2	Il file text_mining.txt viene sovrascritto, col dizionario relativo, e anche gli altri file vengono creati con dizionari non vuoti, e il csv con una riga per ogni file
TCI_2.19	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI4 CFJ2 ESTM1 ESD2 ESF2 ESC2	Il file text_mining.txt viene creato, col dizionario relativo, e anche gli altri file vengono creati con dizionari non vuoti, e il csv con una riga per ogni file

Tabella 2.2: Test case per il modulo Text_Mining

2.1.3 ASA

Per testare gli script e la loro interazione abbiamo individuato i seguenti oggetti dell'ambiente e le relative categorie:

- **Oggetto dell'ambiente 1**: RepositoryMining_ASAResults_neg.csv
  - Categoria: Esistenza (EN)
    - Scelta 1: esiste (EN1) [property negExistOk]
    - Scelta 2: non esiste (EN2)

  - Categoria: Numero record (NN)
    * Scelta 1: Zero (NN1) [if negExistOk]
    * Scelta 2: Uno (NN2) [if negExistOk] [property negNotEmpty]
    * Scelta 3: Maggiore di uno (NN3) [if negExistOk] [property NegNotEmpty] [single]
  - Categoria: Formato contenuto (FCN)
    * Scelta 1: formato valido (FCN1) [if negExistOk] [property negFormatOk]
    * Scelta 2: formato non valido (FCN2) [if negExistOk] [error]
  - Categoria: Tipologia contenuto (TCN)
    * Scelta 1: vulnerabilità assente (TCN1) [if negNotEmpty and negFormatOk]
    * Scelta 2: vulnerabilità presente con regola non ripetuta per componente (TCN2) [if negNotEmpty and negFormatOk]
    * Scelta 3: vulnerabilità presente con regola ripetuta per componente (TCN3) [if negNotEmpty and negFormatOk] [single]

- **Oggetto dell'ambiente 2**: RepositoryMining_ASAResults_pos.csv
  - Categoria: Esistenza (EP)
    * Scelta 1: esiste (EP1) [property posExistOk]
    * Scelta 2: non esiste (EP2)
  - Categoria: Numero record (NP)
    * Scelta 1: zero (NP1) [if posExistOk]
    * Scelta 2: uno (NP2) [if posExistOk] [property posNotEmpty]
    * Scelta 3: maggiore di uno (NP3) [if posExistOk] [property posNotEmpty] [single]
  - Categoria: Formato contenuto (FCP)
    * Scelta 1: formato valido (FCP1) [if posExistOk] [property posFormatOk]
    * Scelta 2: formato non valido (FCP2) [if posExistOk] [error]
  - Categoria: Tipologia contenuto (TCP)
    * Scelta 1: vulnerabilità assente (TCP1) [if posNotEmpty and posFormatOk]
    * Scelta 2: vulnerabilità presente con regola non ripetuta per componente (TCP2) [if posNotEmpty and posFormatOk]

* Scelta 3: vulnerabilità presente con regola ripetuta per componente (TCP3) [if posNotEmpty and posFormatOk] [single]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_3.1	EN2 EP1 NP2 FCP1 TCP2	csv_ASA_final.csv con solo componenti di classe pos
TCI_3.2	EN1 NN2 FCN1 TCN2 EP2	csv_ASA_final.csv con solo componenti di classe neg
TCI_3.3	EN1 FCN2	Index Error
TCI_3.4	EP1 FCP2	Index Error
TCI_3.5	EN1 NN2 FCN1 TCN1 EP1 NP2 FCP1 TCP1	csv_ASA_final.csv con solo header 'Name, class'
TCI_3.6	EN1 NN2 FCN1 TCN2 EP1 NP2 FCP1 TCP2	csv_ASA_final.csv con frequenza = 1 in corrispondenza delle regole non ripetute per componente
TCI_3.7	EN1 NN3 FCN1 TCN3 EP1 NP3 FCP1 TCP3	csv_ASA_final.csv con frequenza > 1 in corrispondenza delle regole ripetute per componente
TCI_3.8	EN1 NN2 FCN1 TCN2 EP1 NP1 FCP1	csv_ASA_final.csv con frequenza = 1 in corrispondenza delle regole ripetute per componente di classe neg
TCI_3.9	EN1 NN1 FCN1 EP1 NP2 FCP1 TCP2	csv_ASA_final.csv con frequenza = 1 in corrispondenza delle regole ripetute per componente di classe pos
TCI_3.10	EN1 NN1 FCN1 EP1 NP1 FCP1	csv_ASA_final.csv con solo header 'Name, class'

Tabella 2.3: Test case per il modulo mining_results_ASA

2.1.4 Union

TotalCombination - Union_TMwithASA

Gli script Union.py e Union_SMwithAsa.py non hanno alcune interazione con altri script all'interno del modulo, al contrario degli script TotalCombination.py e Union_TMwithAsa.py che vengono eseguiti nel seguente ordine:

1. `Union_TMwithAsa.py`
2. `TotalCombination.py`

Per testare gli script TotalCombination.py, Union_TMwithAsa.py e la loro interazione abbiamo individuato i seguenti oggetti dell'ambiente e le relative categorie:

- **Oggetto dell'ambiente 1**: csv_mining_final.csv
  - Categoria: esistenza (EM)
    - Scelta 1: esiste
    - Scelta 2: non esiste [error]
  - Categoria: numero di record (NM)
    - Scelta 1: file contiene zero record
    - Scelta 2: file contiene un record [property notEmptyCSV1][single]
    - Scelta 3: file contiene più di un record [property notEmptyCSV1]
  - Categoria: formato contenuto(FM)
    - Scelta 1: formato valido [if notEmptyCSV1]
    - Scelta 2: formato non valido [if notEmptyCSV1][error]
- **Oggetto dell'ambiente 2**: csv_ASA_final.csv
  - Categoria: esistenza (EA)
    - Scelta 1: esiste
    - Scelta 2: non esiste [error]
  - Categoria: numero di record (NA)
    - Scelta 1: file contiene zero record [property emptyCSV2]
    - Scelta 2: file contiene un record [single]
    - Scelta 3: file contiene più di un record

  - Categoria: formato contenuto(FA)
    - Scelta 1: formato valido [if not emptyCSV2]
    - Scelta 2: formato non valido [error]

- **Oggetto dell'ambiente 3**: Union_TM_ASA.csv
  - Categoria: esistenza (EU)
    - Scelta 1: esiste
    - Scelta 2: non esiste [error]
  - Categoria: numero di record (NU)
    - Scelta 1: file contiene zero record [if emptyCSV1 and emptyCSV2property emptyCSV3]
    - Scelta 2: file contiene un record [single]
    - Scelta 3: file contiene più di un record [if CSV1ok]
  - Categoria: formato contenuto(FU)
    - Scelta 1: formato valido [if not emptyCSV3]
    - Scelta 2: formato non valido [error]

- **Oggetto dell'ambiente 4**: mining_results_sm_final.csv
  - Categoria: esistenza (ES)
    - Scelta 1: esiste
    - Scelta 2: non esiste [error]
  - Categoria: numero di record (NS)
    - Scelta 1: file contiene zero record [property emptyCSV4]
    - Scelta 2: file contiene un record [single]
    - Scelta 3: file contiene più di un record
  - Categoria: formato contenuto(FS)
    - Scelta 1: formato valido [if not emptyCSV4]
    - Scelta 2: formato non valido [error]

Test Case	Combinazione	Oracolo
TC_4.1	EM2	FileNotFoundError
TC_4.2	EA2	FileNotFoundError
TC_4.3	EU2	FileNotFoundError
TC_4.4	ES2	FileNotFoundError
TC_4.5	EM1 NM2 FM1 EA1 NA2 FA1 EU1 NU2 FU1 ES1 NS2 FS1	Il file di Unione conterrà la combinazione delle intestazioni dei file
TC_4.6	EM1 NM3 FM2	UnicodeDecodeError
TC_4.7	EA1 NA3 FA2	UnicodeDecodeError
TC_4.8	EU1 NU3 FU2	UnicodeDecodeError
TC_4.9	ES1 NS3 FS2	UnicodeDecodeError
TC_4.10	EM1 NM1 FM1 EA1 NA1 FA1 EU1 NU1 FU1 ES1 NS1 FS1	Union_TM_ASA.csv viene generato vuoto così come 3COMBINATION.csv
TC_4.11	EM1 NM1 FM1 EA1 NA1 FA1 EU1 NU1 FU1 ES1 NS3 FS1	Union_TM_ASA.csv viene generato vuoto così come 3COMBINATION.csv
TC_4.12	EM1 NM3 FM1 EA1 NA3 FA1 EU1 NU3 FU1 ES1 NS1 FS1	Union_TMwithAsa.py e TotalCombination.py eseguono correttamente l'unione tra i file che conterrà l'unione tra i primi due CSV
TC_4.13	EM1 NM3 FM1 EA1 NA3 FA1 EU1 NU3 FU1 ES1 NS3 FS1	Union_TMwithAsa.py e TotalCombination.py eseguono correttamente, generando i file Union_TM_ASA.csv e 3COMBINATION.csv, entrambi validi e contenenti più di un record.

Test Case	Combinazione	Oracolo
TC_4.14	EM1 NM3 FM1 EA1 NA1 FA1 EU1 NU3 FU1 ES1 NS3 FS1	Union_TMwithAsa.py e TotalCombination.py eseguono correttamente, generando i file Union_TM_ASA.csv e 3COMBINATION.csv, entrambi validi e contenenti più di un record.

Tabella 2.4: Test case per TotalCombination.py, Union_TMwithAsa.py

Union_SMwithAsa.py

Per testare lo script Union_SMwithAsa.py abbiamo individuato i seguenti oggetti dell'ambiente e le relative categorie:

- **Oggetto dell'ambiente 1**: csv_ASA_final.csv
  - Categoria: esistenza (EA)
    - Scelta 1: esiste
    - Scelta 2: non esiste [error]
  - Categoria: numero di record (NA)
    - Scelta 1: file contiene zero record
    - Scelta 2: file contiene un record [property notEmptyCSV1][single]
    - Scelta 3: file contiene più di un record [property notEmptyCSV1]
  - Categoria: formato contenuto(FA)
    - Scelta 1: formato valido [if notEmptyCSV1]
    - Scelta 2: formato non valido [if notEmptyCSV1] [error]
- **Oggetto dell'ambiente 2**: mining_results_sm_final.csv
  - Categoria: esistenza (ES)
    - Scelta 1: esiste
    - Scelta 2: non esiste [error]
  - Categoria: numero di record (NS)
    - Scelta 1: file contiene zero record
    - Scelta 2: file contiene un record [property notEmptyCSV2][single]
    - Scelta 3: file contiene più di un record [property notEmptyCSV2]

  - Categoria: formato contenuto(FS)
    - Scelta 1: formato valido [if notEmptyCSV2]
    - Scelta 2: formato non valido [if notEmptyCSV2][error]

Test Case	Combinazione	Oracolo
TC_5.1	EA2	FileNotFoundError
TC_5.2	EA1 NA1 ES1 NS1	Il file di unione è valido e vuoto
TC_5.3	EA1 NA2 FA1 ES1 NS2 FS1	Il file di Unione è valido, e contiene la combinazione delle due intestazioni
TC_5.4	EA1 NA3 FA2	UnicodeDecodeError
TC_5.5	EA1 NA3 FA1 ES1 NS3 FS2	UnicodeDecodeError
TC_5.6	EA1 NA3 FA1 ES1 NS3 FS1	Il file di Unione è valido, e contiene la combinazione delle intestazioni e dei dati dei due CSV
TC_5.7	EA1 NA1 FA1 ES1 NS3 FS1	Lo script esegue l'unione e rimpiazza i dati mancanti nel file di unione con valori '0'
TC_5.8	EA1 NA3 FA1 ES1 NS1 FS1	Lo script genera un file di output vuoto
TC_5.9	EA1 NA3 FA1 ES2	FileNotFoundError

Tabella 2.5: Test case per Union_SMwithASA.py

Union.py

Per testare lo script Union.py abbiamo individuato i seguenti oggetti dell'ambiente e le relative categorie:

- **Oggetto dell'ambiente 1**: csv_mining_final.csv
  - Categoria: esistenza (ET)
    - Scelta 1: esiste
    - Scelta 2: non esiste [error]
  - Categoria: numero di record (NT)

        * Scelta 1: file contiene zero record
        * Scelta 2: file contiene un record [single] [property notEmptyCSV1]
        * Scelta 3: file contiene più di un record [property notEmptyCSV1]
    - Categoria: formato contenuto(FT)
        * Scelta 1: formato valido [if notEmptyCSV1]
        * Scelta 2: formato non valido [if notEmptyCSV1][error]

- **Oggetto dell'ambiente 2**: mining_results_sm_final.csv
    - Categoria: esistenza (ES)
        * Scelta 1: esiste
        * Scelta 2: non esiste [error]
    - Categoria: numero di record (NS)
        * Scelta 1: file contiene zero record
        * Scelta 2: file contiene un record [property notEmptyCSV2]
        * Scelta 3: file contiene più di un record [property notEmptyCSV2]
    - Categoria: formato contenuto(FS)
        * Scelta 1: formato valido [if notEmptyCSV2]
        * Scelta 2: formato non valido [if notEmptyCSV2] [error]

Test Case	Combinazione	Oracolo
TC_6.1	ET2	FileNotFoundError
TC_6.2	ET1 NT1 ES1 NS1	Il file di unione è valido e vuoto
TC_6.3	ET1 NT2 FT1 ES1 NS2 FS1	Il file di Unione è valido, e contiene la combinazione delle due intestazioni
TC_6.4	ET1 NT3 FT2	UnicodeDecodeError
TC_6.5	ET1 NT3 FT1 ES1 NS3 FS2	UnicodeDecodeError
TC_6.6	ET1 NT3 FT1 ES1 NS3 FS1	Il file di Unione è valido, e contiene la combinazione delle intestazioni

Test Case	Combinazione	Oracolo
TC_6.7	ET1 NT1 FT1 ES1 NS3 FS1	Il file di Unione contiene i dati di mining_results_sm_final.csv
TC_6.8	ET1 NT3 FT1 ES1 NS1 FS1	Lo script genera un file di output vuoto
TC_6.9	ET1 NT3 FT1 ES2	FileNotFoundError

Tabella 2.6: Test case per Union.py