LOG3430 - Méthodes de test et de validation du logiciel

Automne 2020

TP No. 3

Groupe 3

1956925 – Hakim Mektoub

1994973 – Nina Lara Moraga

Soumis à : Humeniuk, Dmytro

8 novembre 2020

Partie 1: Fonctions logiques

Les fonctions sont implémentés dans le fichier email_analyzer.py et on peut voir une capture d'écran de ces fonctions dans l'annexe 1.

Fonction 1:

On a défini les variables suivantes:

S : vrai, si le message est classifié comme Spam.

P : vrai, si le message courant est classifié comme Spam selon le vocabulaire créé.

H : vrai, si le temps entre la date du premier message vu et la date du dernier message vu est est inférieur à un mois.

T1 : vrai, si le niveau de Trust de l'utilisateur est moins que 60.

T2 : vrai, si le niveau de Trust moyen des groupes de l'utilisateur est moins que 70.

T3 : vrai, si le niveau de Trust de l'utilisateur est plus que 75.

La première fonction logique qu'on a est la suivante:

$S = P * (H * T1 + T2) + H * T2 * \neg T3$

On peut dessiner la table de vérité de la fonction ci-dessus afin de nous aider à développer nos tests.

P	**H**	**T1**	**T2**	**T3**	**S(résultat)**
0	0	0	0	0	**0**
0	0	0	0	1	**0**
0	0	0	1	0	**0**
0	0	0	1	1	**0**
0	0	1	0	0	**0**
0	0	1	0	1	**0**
0	0	1	1	0	**0**
0	0	1	1	1	**0**

0	1	0	0	0	**0**
0	1	0	0	1	**0**
0	1	0	1	0	**1**
0	1	0	1	1	**0**
0	1	1	0	0	**0**
0	1	1	0	1	**0**
0	1	1	1	0	**1**
0	1	1	1	1	**0**
1	0	0	0	0	**0**
1	0	0	0	1	**0**
1	0	0	1	0	**1**
1	0	0	1	1	**1**
1	0	1	0	0	**0**
1	0	1	0	1	**0**
1	0	1	1	0	**1**
1	0	1	1	1	**1**
1	1	0	0	0	**0**
1	1	0	0	1	**0**
1	1	0	1	0	**1**
1	1	0	1	1	**1**
1	1	1	0	0	**1**
1	1	1	0	1	**1**
1	1	1	1	0	**1**
1	1	1	1	1	**1**

Active Clause Coverage

On nous demande d'abord de trouver les jeux de tests qui respectent les critères GACC, CACC et RACC.

Pour le critère GACC, la clause majeure doit définir le prédicat, les clauses mineures peuvent changer.

Si P est choisi comme clause majeur on a les jeux de test suivant:

P = 1, H = 0 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 0, H = 0 , T1 = 0, T2 = 1, T3 =0 S = 0

Si H est choisi comme clause majeur, on a les jeux de test suivant:

P = 0, H = 1 , T1 = 1, T2 = 1, T3 =0 S = 1

P = 0, H = 0 , T1 = 1, T2 = 1, T3 =0 S = 0

Si T1 est choisi comme clause majeur, on a les jeux de test suivant:

P = 1, H = 1 , T1 = 1, T2 = 0, T3 =0 S = 1

P = 1, H = 1 , T1 = 0, T2 = 0, T3 =0 S = 0

Si T2 est choisi comme clause majeur, on a les jeux de test suivant:

P = 0, H = 1 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 0, H = 1 , T1 = 0, T2 = 0, T3 =0 S = 0

Finalement, si T3 est choisi comme clause majeur, on a les jeux de test suivant:

P = 0, H = 1 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 0, H = 1 , T1 = 0, T2 = 1, T3 =1 S = 0

On remarque que le résultat est l'inverse de T3, mais un changement de T3, changera le résultat, ce qui fait que c'est T3 qui est déterminant.

Finalement, vu qu'avec le jeu de test qu'on a choisi, les prédicats sont couverts, et que les clauses mineures ne changent pas quand la clause majeure change, on peut dire que notre jeu de test satisfait RACC et CACC aussi.

Notre jeu de test est donc le suivant:
1. P = 1, H = 0 , T1 = 0, T2 = 1, T3 =0 S = 1
2. P = 0, H = 0 , T1 = 0, T2 = 1, T3 =0 S = 0
3. P = 0, H = 1 , T1 = 1, T2 = 1, T3 =0 S = 1
4. P = 0, H = 0 , T1 = 1, T2 = 1, T3 =0 S = 0
5. P = 1, H = 1 , T1 = 1, T2 = 0, T3 =0 S = 1
6. P = 1, H = 1 , T1 = 0, T2 = 0, T3 =0 S = 0
7. P = 0, H = 1 , T1 = 0, T2 = 1, T3 =0 S = 1
8. P = 0, H = 1 , T1 = 0, T2 = 0, T3 =0 S = 0
9. P = 0, H = 1 , T1 = 0, T2 = 1, T3 =1 S = 0

Voir les tests dans l'annexe 2

Inactive Clause Coverage

On nous demande ensuite de déterminer les jeux de tests pour respecter les critères de clauses inactives.

Dans les critères de couverture de clause inactive, on vérifie qu'une clause majeure n'étant pas supposée avoir d'effet sur le prédicat n'en a vraiment pas.

Si P est clause majeure, on peut définir le jeu de test suivant:
P = 1, H = 0 , T1 = 0, T2 = 0, T3 =0 S = 0

P = 0, H = 0 , T1 = 0, T2 = 0, T3 =0 S = 0

P = 0, H = 1 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 1, H = 1 , T1 = 0, T2 = 1, T3 =0 S = 1

Si H est clause majeure, on peut définir le jeu de test suivant:
P = 1, H = 0 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 1, H = 1 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 0, H = 0 , T0 = 0, T2 = 0, T3 =0 S = 0

P = 0, H = 1 , T1 = 0, T2 = 0, T3 =0 S = 0

Si T1 est clause majeure, on peut définir le jeu de test suivant:
P = 0, H = 0 , T1 = 0, T2 = 0, T3 =0 S = 0

P = 0, H = 0 , T1 = 1, T2 = 0, T3 =0 S = 0

P = 0, H = 1 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 0, H = 1 , T1 = 1, T2 = 1, T3 =0 S = 1

Si T2 est clause majeure, on peut définir le jeu de test suivant:
P = 0, H = 0 , T1 = 0, T2 = 0, T3 =0 S = 0

P = 0, H = 0 , T1 = 0, T2 = 1, T3 =0 S = 0

P = 1, H = 1 , T1 = 1, T2 = 0, T3 =0 S = 1

P = 1, H = 1 , T1 = 1, T2 = 1, T3 =0 S = 1

Si T3 est clause majeure, on peut définir le jeu de test suivant:
P = 0, H = 0 , T1 = 0, T2 = 0, T3 =0 S = 0

P = 0, H = 0 , T1 = 0, T2 = 0, T3 =1 S = 0

P = 1, H = 0 , T1 = 0, T2 = 1, T3 =0 S = 1

P = 1, H = 0 , T1 = 0, T2 = 1, T3 =1 S = 1

Encore une fois, on remarque que nos jeux de tests respectent et le critère GICC et le critère RICC, vu que les clauses mineures ne changent pas lorsque la clause majeure change.

On peut voir les tests à l'annexe 3.

Fonction 2

Pour cette fonction, on nous demande de proposer les jeux de tests qui satisfassent les critères IC, PIC et VNS.

La fonction est la suivante : S = P + !T3 * T2:

On a la table de vérité suivante:

P	T3	T2	**S**
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

IC

Pour le critère IC: on peut prendre la négation de la fonction qui nous donnera:
!S = !(P+!T3*T2)
!S = !P * !(!T3*T2)
!S = !P * T3 + !T2
On a donc 4 implicants: !P*T3, !T2, P, !T3*T2
Le jeu de test suivant satisfait le critère IC:
P = faux, T3 = vrai, T2 = faux => S = faux
P = vrai, T3 = faux, T2 = vrai => S = vrai

Les tests sont disponible à l'annexe 4.

PIC

Pour le critère PIC, on veut prendre les implicants premiers, si on prend notre fonction tel que : S = P + !T3*T2, on voit que nos deux implicants soit P et !T3*T2 sont déjà des implicants premiers, donc on n'a pas besoin d'effectuer de manipulations sur la fonction.
La négation de cette fonction est tel que démontré plus haut:
!S = !P * T3 + !T2 On a donc nos 4 implicants premiers : !P*T3, !T2, P, !T3*T2.
On veut créer un jeu de test pour lequel si chaque impliquant est évalué à vrai tous les autres sont évalués à faux.

On produit les jeux de tests suivants
P = faux, T3 = vrai, T2 = vrai => S = faux (!P*T3 est vrai, les autres sont faux)
P = faux, T3 = faux, T2 = faux => S = faux (!T2 est vrai, tous les autres sont faux)
P = vrai, T3 = vrai, T2 = vrai => S = vrai (P est vrai, tous les autres sont faux)
P = faux, T3 = faux, T2 =vrai => S = vrai (!T3*T2 est vrai, tous les autres sont faux)

Les tests sont disponibles à l'annexe 4.

VNS

On a la fonction : S = P + !T3*T2. Nos deux implicants sont P et !T3*T2
Pour P les PUV sont les suivants:
{P = vrai, T3 = vrai, T2 = faux } => S = vrai
{P = vrai, T3 = vrai, T2 = vrai } => S = vrai
{P = vrai, T3 = faux, T2 = faux } => S = vrai

Pour !T3*T2 le PUV est le suivant:
{P = faux, T3 = faux, T2 = vrai } => S = vrai

Pour P les PPF sont les suivants:
{P = faux, T3 = faux, T2 = faux } => S = faux
{P = faux, T3 = vrai, T2 = vrai } => S = faux
{P = faux, T3 = vrai, T2 = faux } => S = faux

Pour !T3*T2 le PPF est le suivant:
{P = faux, T3 = faux, T2 = faux } => S = faux

On a donc le jeu de test suivant:
{P = vrai, T3 = vrai, T2 = faux } => S = vrai
{P = vrai, T3 = vrai, T2 = vrai } => S = vrai
{P = vrai, T3 = faux, T2 = faux } => S = vrai
{P = faux, T3 = faux, T2 = vrai } => S = vrai
{P = faux, T3 = faux, T2 = faux } => S = faux
{P = faux, T3 = vrai, T2 = vrai } => S = faux
{P = faux, T3 = vrai, T2 = faux } => S = faux

Ces tests sont disponible à l'annexe 4

Partie 2 : Tests d'interaction

	CALCULPROBABILITES	COMBINAISONPROBABILITES	CREATIONVOCAB	NETTOYAGETEXTE	DEFINITIONSPAM
1	1	1	1	0	1
2	0	0	1	1	2
3	1	0	1	2	0
4	0	1	1	3	1
5	1	0	2	0	2
6	0	1	2	1	0
7	0	0	2	2	1
8	1	1	2	3	2
9	0	0	3	0	0
10	1	1	3	1	1
11	1	1	3	2	2
12	1	0	3	3	0
13	0	1	4	0	1
14	1	0	4	1	2
15	0	1	4	2	0
16	0	1	4	3	2

Figure : Matrice MCA pour la puissance 2

La taille de la matrice MCA de puissance 2 est de 16 x 5 nous donnant 16 cas de tests.

	CALCULPROBABILITES	COMBINAISONPROBABILITES	CREATIONVOCAB	NETTOYAGETEXTE	DEFINITIONSPAM
1	0	0	1	0	0
2	1	1	1	0	1
3	0	1	1	0	2
4	1	0	1	1	0
5	0	0	1	1	1
6	1	1	1	1	2
7	0	1	1	2	0
8	1	0	1	2	1
9	0	0	1	2	2
10	1	1	1	3	0
11	0	0	1	3	1
12	1	0	1	3	2
13	1	1	2	0	0
14	0	0	2	0	1
15	1	0	2	0	2
16	0	1	2	1	0
17	1	1	2	1	1
18	0	0	2	1	2
19	1	0	2	2	0
20	0	1	2	2	1
21	1	1	2	2	2
22	0	1	2	3	0
23	1	0	2	3	1
24	0	1	2	3	2
25	0	0	3	0	0
26	1	1	3	0	1
27	0	1	3	0	2
28	1	0	3	1	0
29	0	0	3	1	1
30	1	1	3	1	2
31	0	1	3	2	0
32	1	0	3	2	1
33	1	0	3	2	2
34	0	0	3	3	0
35	1	1	3	3	1
36	1	0	3	3	2
37	0	1	4	0	0
38	1	0	4	0	1
39	0	0	4	0	2
40	1	1	4	1	0
41	0	1	4	1	1
42	1	0	4	1	2
43	0	0	4	2	0
44	1	1	4	2	1
45	1	1	4	2	2
46	0	0	4	3	0
47	1	1	4	3	1
48	1	0	4	3	2

Figure : Matrice MCA pour la puissance 3

La taille de la matrice MCA de puissance 3 est de 48 x 5 nous donnant 48 cas de tests.

	CALCULPROBABILITES	COMBINAISONPROBABILITES	CREATIONVOCAB	NETTOYAGETEXTE	DEFINITIONSPAI
43	0	1	2	3	0
44	1	0	2	3	0
45	0	0	2	3	1
46	1	1	2	3	1
47	0	1	2	3	2
48	1	0	2	3	2
49	0	0	3	0	0
50	1	1	3	0	0
51	0	1	3	0	1
52	1	0	3	0	1
53	0	0	3	0	2
54	1	1	3	0	2
55	0	1	3	1	0
56	1	0	3	1	0
57	0	0	3	1	1
58	1	1	3	1	1
59	0	1	3	1	2
60	1	0	3	1	2
61	0	0	3	2	0
62	1	1	3	2	0
63	0	1	3	2	1
64	1	0	3	2	1
65	0	0	3	2	2
66	1	1	3	2	2
67	0	0	3	3	0
68	1	1	3	3	0
69	0	1	3	3	1
70	1	0	3	3	1
71	0	0	3	3	2
72	1	1	3	3	2
73	0	0	4	0	0
74	1	1	4	0	0
75	0	1	4	0	1
76	1	0	4	0	1
77	0	0	4	0	2
78	1	1	4	0	2
79	0	1	4	1	0
80	1	0	4	1	0
81	0	0	4	1	1
82	1	1	4	1	1
83	0	1	4	1	2
84	1	0	4	1	2
85	0	0	4	2	0
86	1	1	4	2	0
87	0	1	4	2	1
88	1	0	4	2	1
89	0	0	4	2	2
90	1	1	4	2	2
91	0	0	4	3	0
92	1	1	4	3	0
93	0	1	4	3	1
94	1	0	4	3	1
95	0	0	4	3	2
96	1	1	4	3	2

Figure : Matrice MCA pour la puissance 4

La taille de la matrice MCA de puissance 4 est de 96 x 5 nous donnant 96 cas de tests.

	CALCULPROBABILITES	COMBINAISONPROBABILITES	CREATIONVOCAB	NETTOYAGETEXTE	DEFINITION:
139	1	0	3	3	1
140	1	1	3	3	1
141	0	0	3	3	2
142	0	1	3	3	2
143	1	0	3	3	2
144	1	1	3	3	2
145	0	0	4	0	0
146	0	1	4	0	0
147	1	0	4	0	0
148	1	1	4	0	0
149	0	0	4	0	1
150	0	1	4	0	1
151	1	0	4	0	1
152	1	1	4	0	1
153	0	0	4	0	2
154	0	1	4	0	2
155	1	0	4	0	2
156	1	1	4	0	2
157	0	0	4	1	0
158	0	1	4	1	0
159	1	0	4	1	0
160	1	1	4	1	0
161	0	0	4	1	1
162	0	1	4	1	1
163	1	0	4	1	1
164	1	1	4	1	1
165	0	0	4	1	2
166	0	1	4	1	2
167	1	0	4	1	2
168	1	1	4	1	2
169	0	0	4	2	0
170	0	1	4	2	0
171	1	0	4	2	0
172	1	1	4	2	0
173	0	0	4	2	1
174	0	1	4	2	1
175	1	0	4	2	1
176	1	1	4	2	1
177	0	0	4	2	2
178	0	1	4	2	2
179	1	0	4	2	2
180	1	1	4	2	2
181	0	0	4	3	0
182	0	1	4	3	0
183	1	0	4	3	0
184	1	1	4	3	0
185	0	0	4	3	1
186	0	1	4	3	1
187	1	0	4	3	1
188	1	1	4	3	1
189	0	0	4	3	2
190	0	1	4	3	2
191	1	0	4	3	2
192	1	1	4	3	2

Figure : Matrice MCA pour la puissance 5

La taille de la matrice MCA de puissance 5 est de 192 x 5 nous donnant 192 cas de tests.

ANNEXE 1

Fonctions à tester pour définir si un message est spam ou pas (Implémentation des fonctions logiques)

```python
@staticmethod
def is_spam_function_one(is_msg_spam, user_historic_in_days, user_trust, user_group_trust):
    p = is_msg_spam
    h = user_historic_in_days < 30
    t1 = user_trust < 60
    t2 = user_group_trust < 70
    t3 = user_trust > 75
    result = p and (h and t1 or t2) or h and t2 and not t3
    return result

@staticmethod
def is_spam_function_two(is_msg_spam, user_trust, user_group_trust):
    p = is_msg_spam
    t2 = user_group_trust < 70
    t3 = user_trust > 75
    result = p or not t3 and t2
    return result
```

ANNEXE II

Tests unitaires pour la fonction 1: Active Clause Coverage

```python
def test_is_spam_function_one_returns_true_acc_test_one(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 35, 65, 60)
    self.assertTrue(return_val)

def test_is_spam_function_one_returns_false_acc_test_two(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 35, 65, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_true_acc_test_three(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 20, 50, 50)
    self.assertTrue(return_val)

def test_is_spam_function_one_returns_false_acc_test_four(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 35, 50, 50)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_true_acc_test_five(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 20, 50, 80)
    self.assertTrue(return_val)

def test_is_spam_function_one_returns_false_acc_test_six(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 20, 70, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_true_acc_test_seven(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 20, 70, 50)
    self.assertTrue(return_val)

def test_is_spam_function_one_returns_false_acc_test_eight(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 20, 70, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_false_acc_test_nine(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 20, 76, 50)
    self.assertFalse(return_val)
```

Annexe III

Tests unitaires pour la fonction 2 satisfaisant l'Inactive Clause Coverage

```python
    # Tests pour l'Inactive Clause Coverage
    # P est clause majeure:
    def test_is_spam_function_one_returns_false_icc_test_one(self):
        return_val = EmailAnalyzer.is_spam_function_one(True, 35, 65, 80)
        self.assertFalse(return_val)

    def test_is_spam_function_one_returns_false_icc_test_two(self):
        return_val = EmailAnalyzer.is_spam_function_one(False, 35, 65, 80)
        self.assertFalse(return_val)

    def test_is_spam_function_one_returns_true_icc_test_three(self):
        return_val = EmailAnalyzer.is_spam_function_one(False, 15, 65, 50)
        self.assertTrue(return_val)

    def test_is_spam_function_one_returns_false_icc_test_four(self):
        return_val = EmailAnalyzer.is_spam_function_one(True, 15, 65, 50)
        self.assertTrue(return_val)

    # H est clause majeure
    def test_is_spam_function_one_returns_true_icc_test_five(self):
        return_val = EmailAnalyzer.is_spam_function_one(True, 20, 65, 60)
        self.assertTrue(return_val)

    def test_is_spam_function_one_returns_true_icc_test_six(self):
        return_val = EmailAnalyzer.is_spam_function_one(True, 15, 65, 60)
        self.assertTrue(return_val)

    def test_is_spam_function_one_returns_false_icc_test_seven(self):
        return_val = EmailAnalyzer.is_spam_function_one(False, 40, 65, 80)
        self.assertFalse(return_val)

    def test_is_spam_function_one_returns_false_icc_test_eight(self):
        return_val = EmailAnalyzer.is_spam_function_one(False, 15, 65, 80)
        self.assertFalse(return_val)
```

```
# T1 clause majeure
def test_is_spam_function_one_returns_false_icc_test_nine(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 35, 65, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_false_icc_test_ten(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 35, 50, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_true_icc_test_eleven(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 20, 50, 60)
    self.assertTrue(return_val)

def test_is_spam_function_one_returns_false_icc_test_twelve(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 20, 65, 60)
    self.assertTrue(return_val)

# T2 clause majeure
def test_is_spam_function_one_returns_false_icc_test_thirteen(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 35, 65, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_false_icc_test_fourteen(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 35, 65, 50)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_true_icc_test_fifteen(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 20, 50, 80)
    self.assertTrue(return_val)

def test_is_spam_function_one_returns_false_icc_test_sixteen(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 20, 50, 50)
    self.assertTrue(return_val)
```

```
# T3 clause majeure
def test_is_spam_function_one_returns_false_icc_test_seventeen(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 35, 65, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_false_icc_test_eighteen(self):
    return_val = EmailAnalyzer.is_spam_function_one(False, 35, 80, 80)
    self.assertFalse(return_val)

def test_is_spam_function_one_returns_true_icc_test_nineteen(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 35, 65, 60)
    self.assertTrue(return_val)

def test_is_spam_function_one_returns_false_icc_test_twenty(self):
    return_val = EmailAnalyzer.is_spam_function_one(True, 35, 80, 60)
    self.assertTrue(return_val)
```

Annexe IV

Tests unitaires pour la fonction 2 qui satisfassent le critère IC

```python
# Critère IC
def test_is_spam_function_two_returns_false_ic_test_one(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 80, 80)
    self.assertFalse(return_val)

def test_is_spam_function_two_returns_true_ic_test_two(self):
    return_val = EmailAnalyzer.is_spam_function_two(True, 20, 65)
    self.assertTrue(return_val)
```

Tests unitaires pour la fonction 2 qui satisfassent le critère PIC

```python
# Critère PIC
def test_is_spam_function_two_returns_false_pic_test_one(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 80, 65)
    self.assertFalse(return_val)

def test_is_spam_function_two_returns_false_pic_test_two(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 20, 80)
    self.assertFalse(return_val)

def test_is_spam_function_two_returns_true_pic_test_three(self):
    return_val = EmailAnalyzer.is_spam_function_two(True, 80, 65)
    self.assertTrue(return_val)

def test_is_spam_function_two_returns_true_pic_test_four(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 20, 50)
    self.assertTrue(return_val)
```

Tests unitaires pour la fonction 2 qui satisfont le critère VNS.

```python
# Critère VNS
def test_is_spam_function_two_returns_true_vns_test_one(self):
    return_val = EmailAnalyzer.is_spam_function_two(True, 80, 80)
    self.assertTrue(return_val)

def test_is_spam_function_two_returns_true_vns_test_two(self):
    return_val = EmailAnalyzer.is_spam_function_two(True, 80, 65)
    self.assertTrue(return_val)

def test_is_spam_function_two_returns_true_vns_test_three(self):
    return_val = EmailAnalyzer.is_spam_function_two(True, 20, 80)
    self.assertTrue(return_val)

def test_is_spam_function_two_returns_true_vns_test_four(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 20, 65)
    self.assertTrue(return_val)

def test_is_spam_function_two_returns_false_vns_test_five(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 20, 80)
    self.assertFalse(return_val)

def test_is_spam_function_two_returns_false_vns_test_six(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 80, 60)
    self.assertFalse(return_val)

def test_is_spam_function_two_returns_false_vns_test_seven(self):
    return_val = EmailAnalyzer.is_spam_function_two(False, 80, 80)
    self.assertFalse(return_val)
```