UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DOCUMENTAZIONE DEL PROGETTO DI INGEGNERIA DELLA CONOSCENZA
2022/2023

SP-AI-WARE
Laboratorio di analisi Malware con algoritmi di Machine Learning e supporto al Web Semantico

Giuseppe Emanuele De Musso
Matricola: 706261
Email: g.demusso5@studenti.uniba.it

URL Repository: https://github.com/pr4gm4/Sp-AI-Ware.git

INDICE

INTRODUZIONE

Sp-Ai-Ware si propone di essere un laboratorio di analisi Malware potenziato da algoritmi di Machine Learning e con supporto al Web Semantico.

Il suo funzionamento e' semplice. La piattaforma prende in input il valore sha di un software e le sue Features binarie. In ambito di Antivirus e Sicurezza informatica in generale, il valore sha di un software non e' altro che un file eseguibile su cui viene applicato un algoritmo di hashing per poi produrre una stringa di lunghezza prefissata che identifica univocamente il software in modo che , se un utente segnala che un certo software e' malevolo, il suo sha (che in termini piu' generici si chiama firma) viene caricato su un database. L'utente che utilizza un Antivirus ha quindi (tra le altre cose) una lista sempre aggiornata di firme riguardanti software malevoli in modo che , se l'utente dovesse inavvertitamente scaricare il software la cui firma e' presente nel database, esso verrebbe bloccato.

Sp-AI-Ware utilizza un classificatore che analizza le caratteristiche binarie del programma e , qualora l'esito della classificazione dovesse essere positivo, procedera' a caricare la firma su una base di conoscenza OWL/RDF in modo da fornire informazioni condivise utili all'essere umano e comprensibili alla macchina. In piu' possiede un golossario che estrae delle informazioni inerenti alla Cybersecurity da Dbpedia.org

STRUMENTI UTILIZZATI

Linguaggio di programmazione Python: Python oltre a essere un linguaggio di programmazione semplice, e' ricco di librerie utilissime nel campo dell'intelligenza artificiale e dell'ingegneria della conoscenza.

Protége: questo software fornisce un'intuitiva interfaccia grafica per poter realizzare delle basi di conoscenza basate sul Web semantico con sintassi OWL/RDF. In piu' implementa dei Reasoner come Hermitt e Pellet per poter verificare la consistenza dell'ontologia e inferire nuove regole a partire da quelle esistenti.

Pandas, Numpy, ScikitLearn, Matplotlib: queste 4 librerie Python sono tra le piu' famose per quanto riguarda il campo dell' IA e permettono con facilita' di manipolare grandi quantita' di dati, visualizzarli graficamente, e creare modelli predittivi facilmente.

Owlready2: libreria utile a manipolare programmaticamente basi di conoscenza OWL/RDF. Anche questa libreria implementa un interfaccia con il reasoner Hermitt (che e' quello usato in tutto il progetto)

SparqlWrapper: utile libreria che fornisce un interfaccia tra Python e un endpoint Sparql (in questo caso con quello di Dbpedia)
Visual Studio Code: editor di codice altamente flessibile con molte estensioni per poter facilmente lavorare anche con sintassi OWL/RDF e file Jupyter Notebook

DESCRIZIONE DATASET

Il dataset utilizzato (con rif bibliografico alla fine del documento) e' molto grande e le sue Features sono proprieta' di file eseguibili che sono di dominio estremamente tecnico e difficili da comprendere singolarmente , tuttavia e' possibile descrivere le parti fondamentali del dataset.

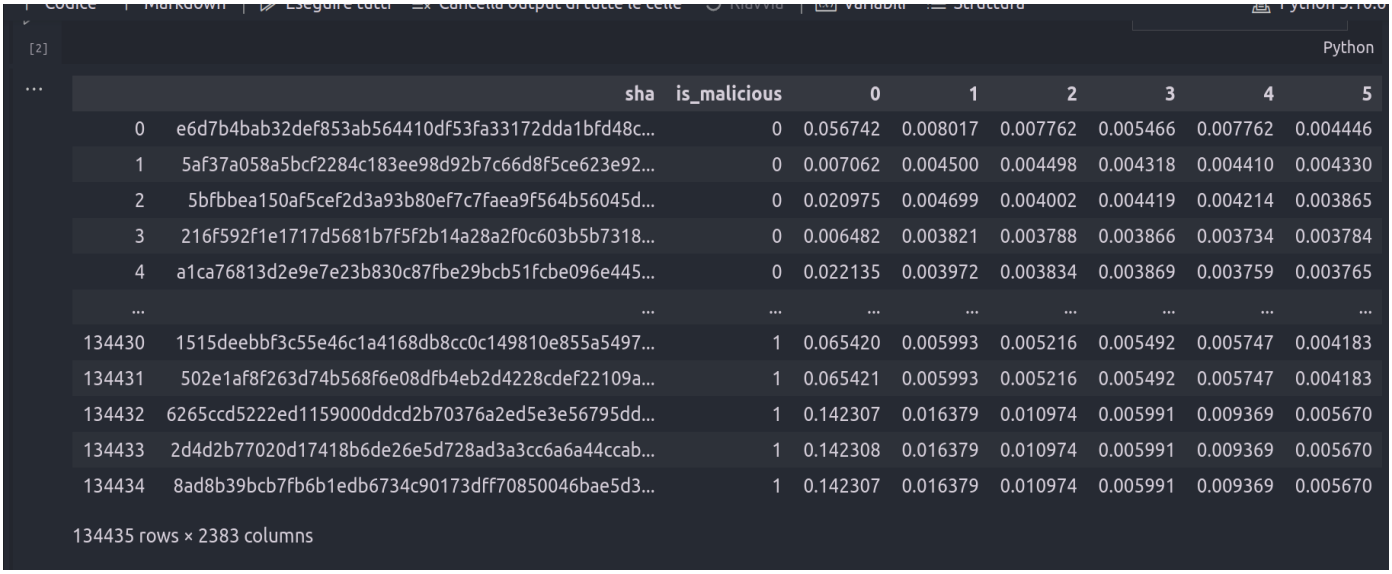
FEATURES(X):

sha: come detto in precedenza, e' una stringa di lunghezza fissa che rappresenta univocamente l'eseguibile all'interno del dataset

features da 0 a 2380: queste feature rappresentano le proprieta' che il file eseguibile possiede. Sono caratteristiche utilizzate in ambito di analisi malware che stabiliscono ad esempio, la dimensione del Portable Executable Header, numero delle Syscalls, tentativi di Stack Smashing rilevati nell'esecuzione … ecc ecc)

TARGET(Y):

is_malicious: intuitivamente, questa variabile assume valore 1 se il software e' malevolo, 0 se si tratta di un software benigno



[2] Python

	sha	is_malicious	0	1	2	3	4	5
0	e6d7b4bab32def853ab564410df53fa33172dda1bfd48c...	0	0.056742	0.008017	0.007762	0.005466	0.007762	0.004446
1	5af37a058a5bcf2284c183ee98d92b7c66d8f5ce623e92...	0	0.007062	0.004500	0.004498	0.004318	0.004410	0.004330
2	5bfbbea150af5cef2d3a93b80ef7c7faea9f564b56045d...	0	0.020975	0.004699	0.004002	0.004419	0.004214	0.003865
3	216f592f1e1717d5681b7f5f2b14a28a2f0c603b5b7318...	0	0.006482	0.003821	0.003788	0.003866	0.003734	0.003784
4	a1ca76813d2e9e7e23b830c87fbe29bcb51fcbe096e445...	0	0.022135	0.003972	0.003834	0.003869	0.003759	0.003765
...	...	...	...	...	...	...	...	...
134430	1515deebbf3c55e46c1a4168db8cc0c149810e855a5497...	1	0.065420	0.005993	0.005216	0.005492	0.005747	0.004183
134431	502e1af8f263d74b568f6e08dfb4eb2d4228cdef22109a...	1	0.065421	0.005993	0.005216	0.005492	0.005747	0.004183
134432	6265ccd5222ed1159000ddcd2b70376a2ed5e3e56795dd...	1	0.142307	0.016379	0.010974	0.005991	0.009369	0.005670
134433	2d4d2b77020d17418b6de26e5d728ad3a3cc6a6a44ccab...	1	0.142308	0.016379	0.010974	0.005991	0.009369	0.005670
134434	8ad8b39bcb7fb6b1edb6734c90173dff70850046bae5d3...	1	0.142307	0.016379	0.010974	0.005991	0.009369	0.005670

134435 rows × 2383 columns

Come e’ possibile notare, il dataset e’ troppo grande e richiederebbe una potenza computazionale non indifferente.

E’ necessario quindi effettuare un sampling e utilizzare un dataset piu’ piccolo . Nel progetto quindi E’ stato piu’ idoneo utilizzare un dataset con 5000 osservazioni prese randomicamente a partire dal dataset originale .

DATASET REFINEMENT

Dopo il sampling del dataset originale ci sono altri 2 problemi che potrebbero portare a dei risultati inaffidabili

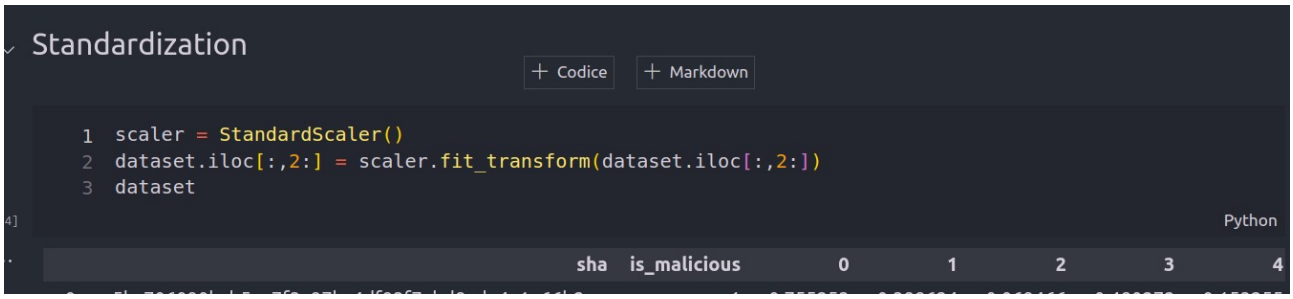
1) il dataset non e' normalizzato e/o standardizzato

2) le features sono comunque molte per un tempo computazionale accettabile

Si passa quindi a risolvere queste problematiche.

NORMALIZZAZIONE: e' il processo di scalare i valori delle Feature a un range comune in modo che i modelli possano performare meglio e l'impatto degli outlier sulla predizione finale sara' minore. Tra le tecniche piu' famose, non che quella utilizzata, troviamo la tecnica MinMax che scala i valori delle features in un range che va da 0 a 1 .

STANDARDIZZAZIONE: e’ il processo di trasformare i valori delle features in modo che la media di ogni feature e’ 0 e la deviazione standard e’ 1 . Anche questo come la normalizzazione , serve a far performare meglio il modello nelle predizioni



NB: nel corso dei due processi , sono stati ovviamente tolte le caratteristiche come sha il cui valore e’ solo identificativo dell’eseguibile

FEATURE FILTERING: il dataset presenta comunque troppe features e non e' detto che tutte contribuiscano alla predizione finale in egual misura, percio' e' importante trovare quali sono le features che contribuiscono di piu' al risultato finale . Nel progetto viene utilizzata la correlazione di Spearman e vengono selezionate le 25 features piu' correlate alla variabile target "is_malicious"

Feature Filtering

```
1 correlation = dataset.corr("spearman")
2 correlation
```

[5] Python

	is_malicious	0	1	2	3	4	5	6	7	8	...	2371	2372	2373	2374	2375	2376	2377	2378
is_malicious	1.000000	0.008487	-0.237635	-0.217832	-0.198030	0.025461	-0.178227	-0.099015	-0.043850	-0.121647	...	-0.300760	-0.300715	NaN	NaN	-0.489883	-0.399192	-0.087185	-0.087185
0	0.008487	1.000000	0.490516	0.206819	0.088884	0.217191	-0.192797	-0.190108	-0.441115	0.212965	...	0.173743	0.165295	NaN	NaN	-0.111106	-0.107554	0.153499	0.153499
1	-0.237635	0.490516	1.000000	0.760288	0.687971	0.714766	0.342905	0.431741	0.148959	0.748571	...	0.354379	0.353304	NaN	NaN	0.048426	0.064371	0.290056	0.290056
2	-0.217832	0.206819	0.760288	1.000000	0.825978	0.725618	0.505594	0.542377	0.328363	0.684706	...	0.144985	0.157639	NaN	NaN	0.027347	0.118628	0.303527	0.303527
3	-0.198030	0.088884	0.687971	0.825978	1.000000	0.702185	0.575030	0.567443	0.496050	0.595966	...	0.188803	0.190902	NaN	NaN	0.043067	0.058758	0.185137	0.185137
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2376	-0.399192	-0.107554	0.064371	0.118628	0.058758	-0.195084	0.091575	-0.074332	0.019898	0.072006	...	0.319058	0.331238	NaN	NaN	0.898864	1.000000	0.480085	0.480085
2377	-0.087185	0.153499	0.290056	0.303527	0.185137	0.116451	-0.083281	-0.170645	-0.102471	0.386503	...	0.348931	0.380249	NaN	NaN	0.426845	0.480085	1.000000	1.000000
2378	-0.087185	0.153499	0.290056	0.303527	0.185137	0.116451	-0.083281	-0.170645	-0.102471	0.386503	...	0.348931	0.380249	NaN	NaN	0.426845	0.480085	1.000000	1.000000
2379	-0.329435	0.385435	0.433365	0.469312	0.277593	0.369459	0.165757	0.361470	0.021968	0.197710	...	-0.148621	-0.148599	NaN	NaN	-0.233352	-0.145107	-0.118849	-0.118849
2380	-0.328237	0.377349	0.422876	0.463548	0.264725	0.362146	0.163961	0.360157	0.006925	0.186645	...	-0.148081	-0.148059	NaN	NaN	-0.239039	-0.144831	-0.118417	-0.118417

2382 rows × 2382 columns

```
1 best_features = correlation.iloc[0]
2 best_features = best_features.abs()
3 best_features = best_features.sort_values(ascending=False)
4 best_features = best_features[0:26]
5 best_index = best_features.index
6 best_index
```

[6] Python

```
Index(['is_malicious', '658', '2360', '2359', '315', '316', '886', '613',
       '312', '655', '734', '834', '1416', '317', '846', '2375', '677', '968',
       '618', '626', '318', '679', '2363', '882', '2364', '725'],
      dtype='object')
```

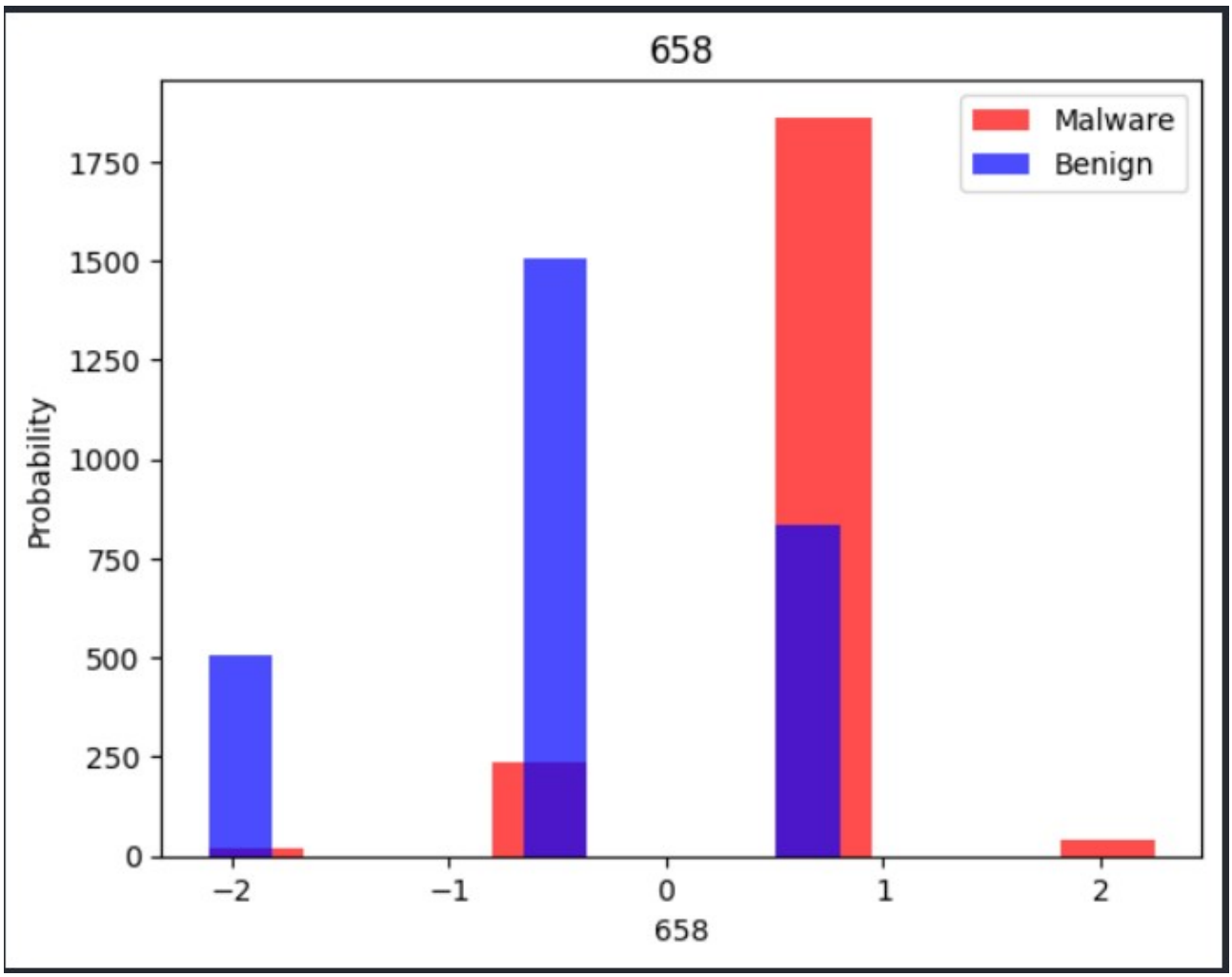
```
1 ref_dataset = dataset.loc[:,best_index]
2 ref_dataset["sha"] = dataset.loc[:,"sha"]
3 ref_dataset
```

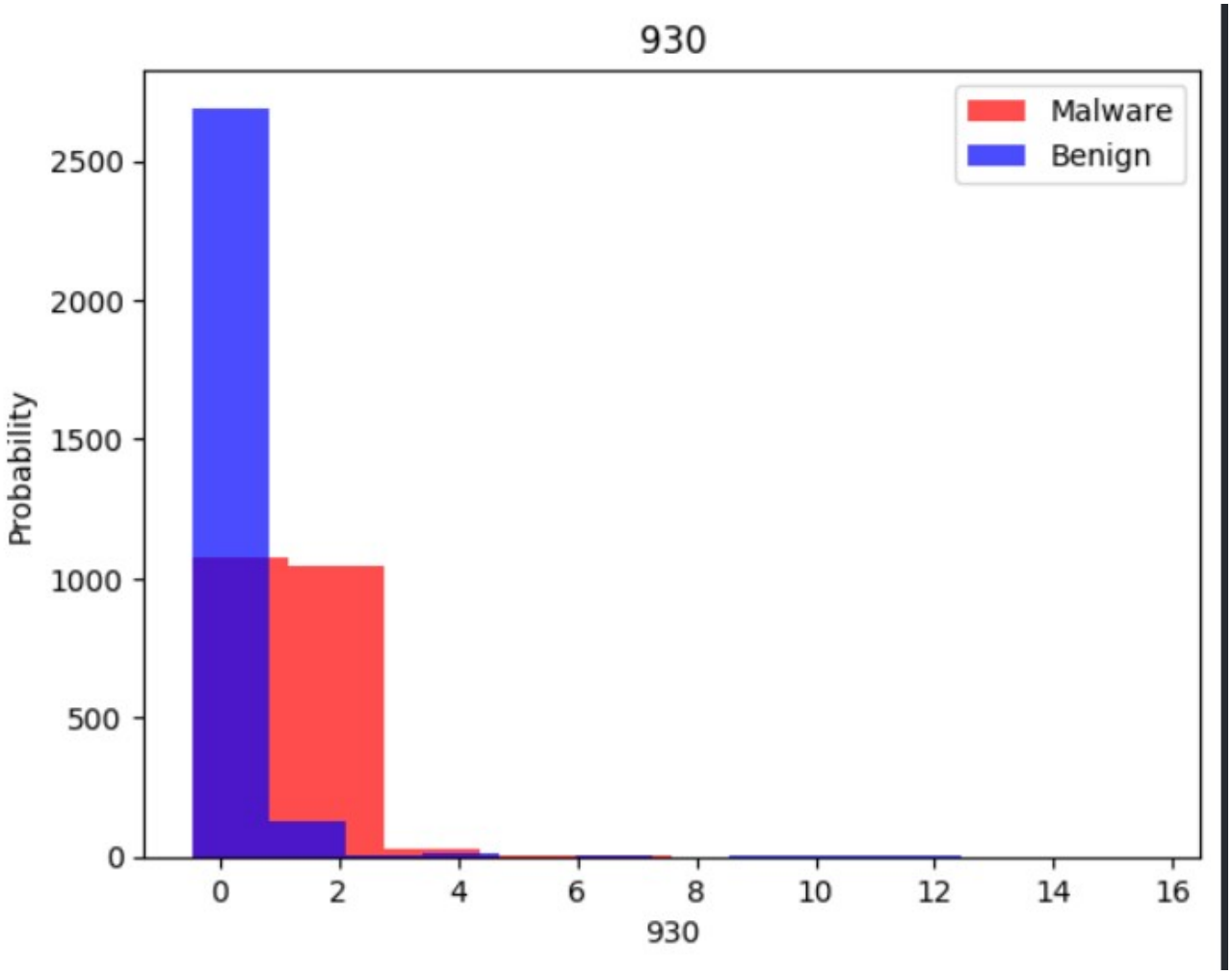
[7] Python

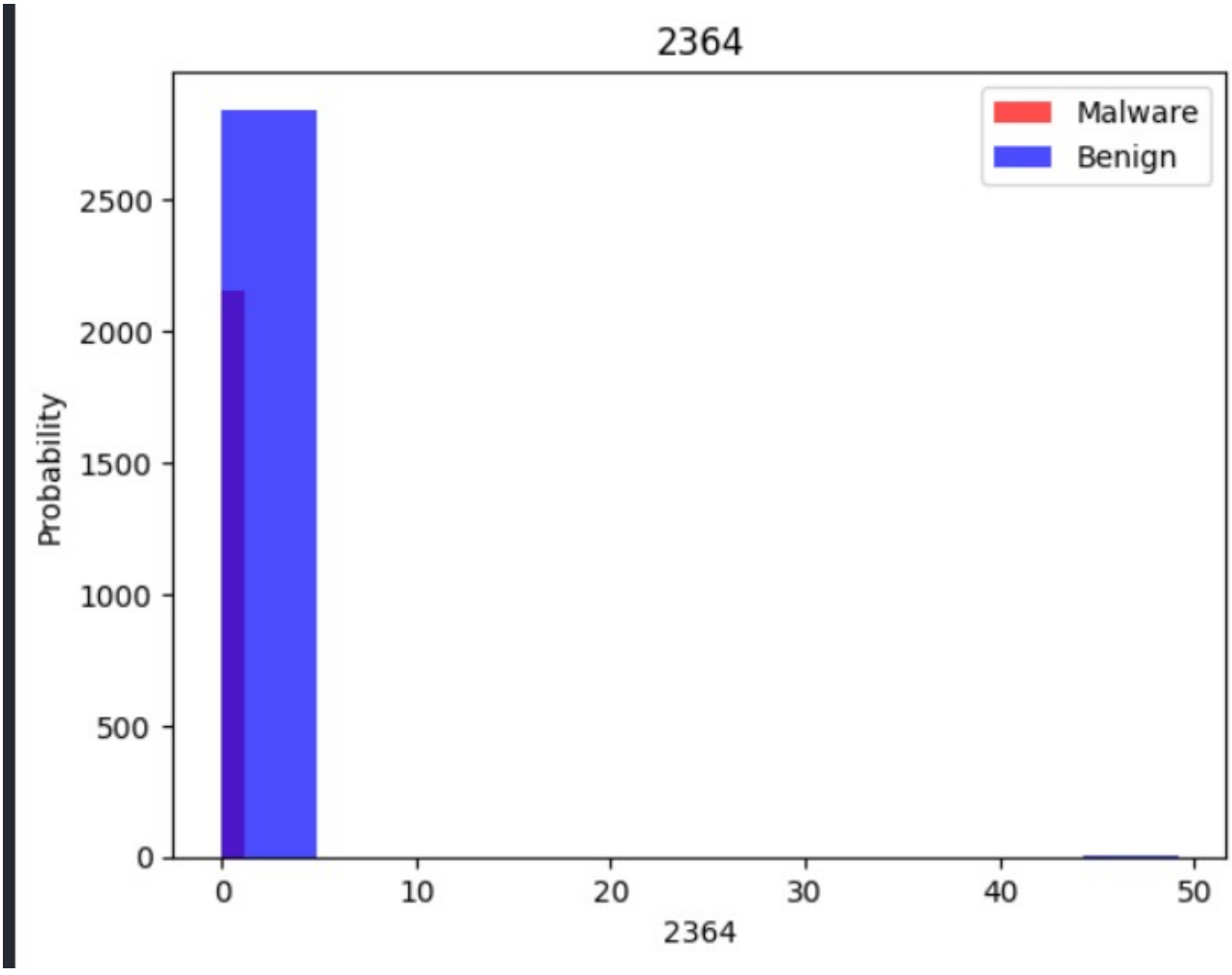
NB: negli indici finali, “is_malicious” e’ la feature che contribuisce di piu’ a se stessa, quindi viene scartata rendendo la piu’ correlata la 658 2360 e cosi via

ANALISI DEI DATI

dopo aver rifinito il dataset , e’ possibile analizzare i dati disponibili e dare un’interpretazione significativa per la scelta del modello o dei modelli da utilizzare successivamente . In questo progetto viene utilizzato un istogramma dove, sull’asse delle ascisse ci saranno tutte le features del dataset e sull’asse delle ordinate ci sara’ la densita’ delle osservazioni . In uno stesso grafico verranno sovrapposti gli istogrammi degli eseguibili malevoli (colorati di rosso) e degli eseguibili benigni (colorati di blu) in modo che e’ possibile notare la differenza in ogni feature e per quali valori . Ecco alcuni esempi :


658
Malware
Benign
Probability
1750
1500
1250
1000
750
500
250
0
−2
−1
0
1
2
658


930
Malware
Benign
Probability
2500
2000
1500
1000
500
0
0
2
4
6
8
10
12
14
16
930


2364
Malware
Benign
Probability
2500
2000
1500
1000
500
0
0
10
20
30
40
50
2364

Come e’ possibile notare, i dati tra software benigni e malevoli sono ben differenziati tra loro. Questo assieme alla variabile da predire (che e’ di tipo booleano) ha portato alla scelta di 4 modelli di classificazione uniti attraverso la tecnica Stacking.

MODELLI PER LA CLASSIFICAZIONE

Con l’analisi dei dati fatta in precedenza , e’ stato possibile notare che i dati tra software malevoli e benigni sono molto eterogenei e che l’esito finale e’ di tipo booleano. Questo ha portato alla scelta di 4 modelli uniti attraverso la tecnica di stacking. Questi modelli sono Albero di decisione,Random Forest, Regressione Logistica e Support Vector Machine.

Albero di decisione: l’utilizzo di questo modello sul dataset considerando le feature progressivamente piu’ correlate alla variabile target, comporta uno split del dataset efficate che porta il modello a terminare la sua predizione in pochi nodi di split

Random Forest: stesso discorso solo che vengono effettuati training multipli (Bagging) e su campioni scelti randomicamente (Boosting)

Regressione Logistica e Support Vector Machine: efficaci nel caso in cui la classificazione e’ di tipo binaria

i modelli allenati saranno dati in input al classificatore a Stack che terra’ conto delle predizioni fatte dai 4 e attraverso un meta-modello (di default Regressione Logistica) decidera’ l’esito finale.

Il traning sara’ eseguito sia suddividendo il dataset in training e test set , sia utilizzando la tecnica di k-fold validation per evitare l’overfitting o comunque predizioni imprecise. Il parametro k e’ settato a 10 fold.

METRICHE E PRESTAZIONI

qui di seguito vengono illustrate le principali metriche per ognuno dei 4 modelli e del modello stack senza tecnica k-fold e poi la metrica “accuracy” per la predizione con k-fold

Dataset setup

```
1  X = dataframe.iloc[:,1:-1]
2  y = dataframe.iloc[:,:1]
3  X = X.values
4  y = y.values
5  X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
6
7
8
```

[16] Python

Decision Trees

```
1  from sklearn.tree import DecisionTreeClassifier
2
3  decision_tree_model = DecisionTreeClassifier()
4  decision_tree_model.fit(X_train,np.ravel(y_train))
5  y_predicted = decision_tree_model.predict(X_test)
6
7  #Performance
8  print(classification_report(y_predicted,y_test))
9
10
```

[17] Python

```
...               precision    recall  f1-score   support

           0       0.95      0.97      0.96       546
           1       0.96      0.94      0.95       454

    accuracy                           0.95      1000
   macro avg       0.95      0.95      0.95      1000
weighted avg       0.95      0.95      0.95      1000
```

Logistic Regression

```python
from sklearn.linear_model import LogisticRegression

logistic_model = LogisticRegression()
logistic_model.fit(X_train,np.ravel(y_train))
y_predicted = logistic_model.predict(X_test)


#Performance
print(classification_report(y_predicted,y_test))
```

[18] Python

```
              precision    recall  f1-score   support

           0       0.89      0.92      0.90       540
           1       0.90      0.87      0.88       460

    accuracy                           0.89      1000
   macro avg       0.89      0.89      0.89      1000
weighted avg       0.89      0.89      0.89      1000
```

Support Vector Classifier

```python
from sklearn.svm import SVC

svc_model = SVC(C=3)
svc_model.fit(X_train,np.ravel(y_train))
y_predicted = svc_model.predict(X_test)

#Performance
print(classification_report(y_predicted,y_test))
```

[19] Python

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       548
           1       0.95      0.94      0.94       452

    accuracy                           0.95      1000
   macro avg       0.95      0.95      0.95      1000
weighted avg       0.95      0.95      0.95      1000
```

Random Forest

```python
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier()
rf_model.fit(X_train,np.ravel(y_train))
y_predicted = rf_model.predict(X_test)

#Performance
print(classification_report(y_predicted,y_test))
```

[20] Python

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       559
           1       0.96      0.97      0.96       441

    accuracy                           0.97      1000
   macro avg       0.97      0.97      0.97      1000
weighted avg       0.97      0.97      0.97      1000
```

Stacking

```python
from sklearn.ensemble import StackingClassifier

estimators = [
    ("decision tees",decision_tree_model),
    ("logistic Regression",logistic_model),
    ("random forest",rf_model),
    ("svc",svc_model),

]

stacking_model = StackingClassifier(estimators=estimators)

stacking_model.fit(X_train,np.ravel(y_train))
y_predicted = stacking_model.predict(X_test)
print(classification_report(y_predicted, y_test))
```

[21] Python

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       562
           1       0.96      0.97      0.96       438

    accuracy                           0.97      1000
   macro avg       0.97      0.97      0.97      1000
weighted avg       0.97      0.97      0.97      1000
```

qui di seguito lo score della tecnica k-fold sul training e test set

INTERFACCIA E ONTOLOGIA OWL/RDF

Sp-AI-Ware presenta un’interfaccia a riga di comando dove e’ possibile scansionare un software da una lista di 50 gia’ campionati o avere una descrizione delle varie classi dell’ontologia formulando una query Sparql a Dbpedia che restituira’ una descrizione della classe indicata . Nel caso in cui l’utente scelga la prima opzione si presentera’ una lista con tutti gli sha rappresentativi dei programmi da scansionare

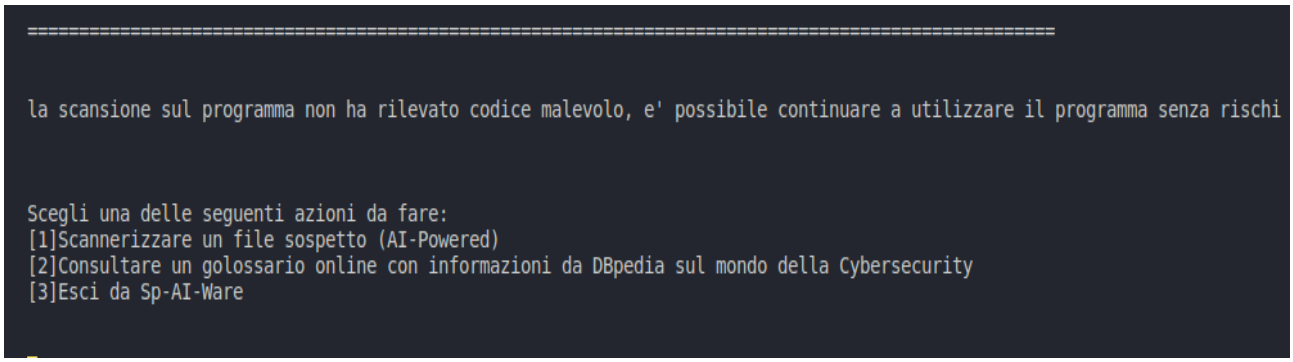
```
programma->0 5bc706090bcb5aa7f3a97bc4df93f7abd8ade4a1e66b2c168d9f2c74bd667b9e
programma->1 520486c40b6d918f2ca60cf5c0fbc2e73aa5293d2d6399bc977d16c116fbe828
programma->2 5c2678a67e9764422d025dc7e43371ee69aacccf1bddddb03ed4728c5a42c977
programma->3 b057b7cf1375c60d69a1b42bfb27e586cdb3992f7be65af01560697238428ddf
programma->4 90e6dbdb4eb72ed9bcb4dcdc53f9a3ce3108297f84ce346b9af04790a641ccce
programma->5 76a05e87593b5bdd0a73d9acd0cbc9e3e809ebf7a674888589f46b73e0350fd7
programma->6 a4848885d94ad0e801f07e3a023f4b0b1fc009bc0486dd69a8b86f3ca0e82bf5
programma->7 de51942a5389bcf2e5df0ad0d8f66fe1f9c7d9e9969df8aff840a6fba8a865f1
programma->8 3cbca875147abe882473fe3709dd270863bb1aa5e26887088f18ed84ebb3d4b8
programma->9 d15dcc0dfe144b7fd40a0a07a9bc47f8f1629f4c0503b0723d26438ba646b3f8
programma->10 d5b26b2f9f4af0d7cc09eb21f6dc8b7ae2930eaeac4bda41e85f3d0b3ff649f7
programma->11 741bb7a81f9910946cb073814599f2ba195678d6fd6ce57dd2718f07c42665dc
programma->12 68a713e79bd17feeb09b74c35ae8e1323d67c057f47d573f3d06dd65b9058d40
programma->13 917510c4ed80acbeea7e5283d48d3f90a0bdaefbe1d812a1010d88598f122249
programma->14 d2b9369e693390c02dd127d17f837b38c5ad42f685aa498e776f4e69e175eb3a
programma->15 20b82fbc859ca16f8250852354e35b164e49c7b3b058138f677fc07d115b78ee
programma->16 d2b69bbc0eafcb917a3dbdc5536701ccb9828dcd4c9b2ee5d40275134096d943
programma->17 a132885c6be273499501ab3dcc6949177f92241955e810ab5f9916d32cc7d236
programma->18 54b26a67e4dea1f9c58c6cb1a5e6d82127df9b6446288f260656d4620ed7a8da
programma->19 740285983c4472b74198c638e3b51f97f3b3220dd18f6f459ef4a31d04ed63ea
programma->20 9fcba84e70c2a39ac122b9961217cf2e23ab60467e451715c58e911edafe3df9
programma->21 08fb552616765779 95a9d7bc89bdbdfc6261d1de5a9499c562deced82e8055bd
programma->22 a32807f66240fbc5955595bce38bebbb4c169c8c112c8fd2a3a0cd2b6bbbf93c
programma->23 4f79cdc15336fac9911450a6dd80edfa6339a10e84e5e7dd6a2ef0ee31b81248
programma->24 8893385a4ad7391c51cb4956efb024167e777074770f47f8ba4660496a493dc9
programma->25 967389f4758d247606008a164265cfa57ef251524137fd03221a1ab2cfd93ca4
programma->26 2ada68f81b79e4948e1638284b15d189f1c84539af60e103b2d5294893401711
programma->27 ed6a47031574460ea0eb9a6ba936ba42314215c40b3876d8c7b39f7a9b6e9b5c
programma->28 e6ecc4585e3aa9acd79050fa3832b8e589ae52e3fb248eeea8a6e59aa907ad68
programma->29 7ba417fe38dbfcfe3e338350fc5ecaa40b4f56a0dd216f8a083de15c45478aa0
programma->30 41d61955a51906ee211b8c2ec0468c7655c286a78caa9584b334d043d071d482
programma->31 89f3c093106bdcbc737c54ef80c0c961097115271a9a73c91c9fbcb41a2d6ec3
programma->32 e3cfa3dcea492f651a4176bdc1ca6d13d54ec7f9c0612d339d912a3547005e14
programma->33 3b6678c397d627e92ee0db57fa33ba18f8c52003c2a7e4ccd1b9508c16c1a5fe
programma->34 aebbb1a671a7c68d4e6e86f4ee6ffced681933648382e5980913cbd864d02fe7
programma->35 aaf79a2ff9e75e9a6b0d1730dca10fdf9018a44f86d17595e0b97b557d611875
programma->36 38f0508164f0f8168a94d0d37affe3d3c95debc301938fcd176ff8d5736a5174
programma->37 c8128fb683d1f271cbc0dff077d18b902daeb5489f62e8fc58df2348e62321c3
programma->38 4922358261b368798b11ecb96bda7519d241fea919c0111e763cfb3fe028f7f4
programma->39 c55a4bf4a50e3e503d5dd66d60f31afc90325fc6211c72126613ab2be88785c1
programma->40 0160cb3fcb66a62e08163beda20b182b5bccd7f319f1c63a8f1d6bb8a6c1d0a8
programma->41 f38ea7112fea4e469883ef0935c9b47e31386e6a992494eac12f1067bb45e86a
programma->42 eec214d9bcde80e26c3f5314d9041107a2c0cd978d62620a64152fd33ed35267
programma->43 b51f3cdd4ca07283ed8f6212f0be3a011dc42f565d38cc96413ca95b31fb92bd
programma->44 b878772a9364bbfefbfabe367b64886ae5baf4773e66a0907d7f0c2612cdab83
programma->45 cc2698ae68ad6e660e9e7a3ec14c216e63b683f57321e4bcf781913a23cb7afe
programma->46 4bf1f29c27a3f6cf2f31f0541adaff497cea441e8f83fa412081bfc1b4523983
programma->47 08bb7bda3ceb55ef6bded90a45fb8fd847409133db59732da9c4c67fd474827c
programma->48 9b41245c4844b28d20aef31a0fd6e4dca5b6d00cb3b69954bd884a661c6edf84
programma->49 2aa6036fd06c28054c347f1b97a6d1a9b7387b8a1cb5e42bbd3e0a8f84856d9e
```

NB: i programmi sono scelti randomicamente dal pool principale, quindi la proporzione tra programmi benigni e maligni non e' nota. In ogni caso e' assai improbabile che non ci sia neanche 1 file maligno tra i 50 scelti

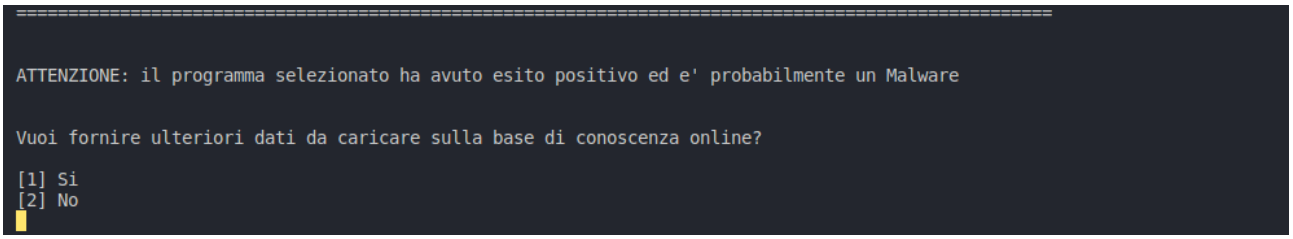
NB2: il dataset dei 50 elementi manca della variabile target "is_malicious" perche' l'intento e' di emulare un sistema che deve scoprire sul momento la natura di un eseguibile piuttosto che avere gia' una risposta

L’eseguibile scelto dall’utente viene classificato dal modello (che , dopo il suo training e test vari , viene salvato nella directory Model e viene caricato e usato al momento in cui l’utente avvia il porgramma).

Se l’esito dovesse essere negativo, l’utente viene rimandato al menu’ principale per una scelta o per terminare il programma.

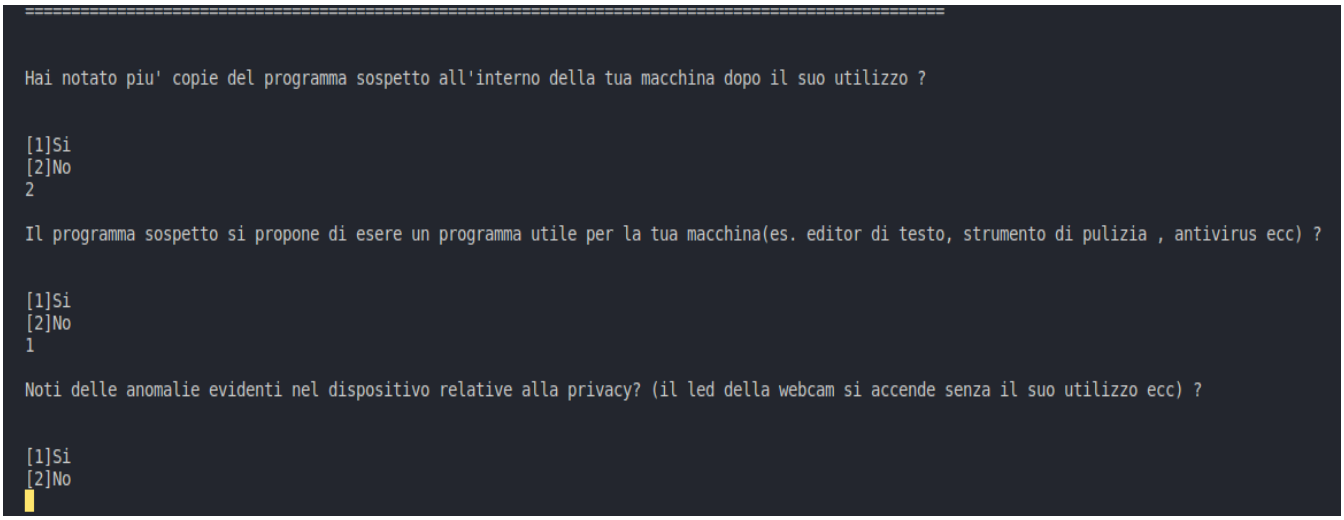


Se invece l’esito dovesse essere positivo , l’utente verrebbe allertato dal software, chiedendo di poter inserire piu’ informazioni per meglio identificare la natura del Malware



Nel caso il cui l’utente sceglie di inserire piu’ informazioni, il sistema formulera’ una serie di domande per classificare il Malware in 11 categorie distinte : Adware, Backdoor, Keylogger, Ransomware, Rogue, Rootkit, Spyware, Trojan, Virus, Wiper e Worm. Nel caso in cui le risposte non dovessero identificare il Malware, esso verra’ classificato come Grayware(cioe’ un software che non ha necessariamenti intenti malevoli ma peggiora le prestazioni del sistema e/o non si conoscono informazioni certe sul suo funzionamento.

NB: in questo software e nella Sicurezza informatica in generale, non e’ sempre possibile categorizzare un Malware in un unica categoria perche’ spesso un Malware e’ una combinazione di piu’ tecniche, pertanto nell’ontologia OWL/RDF non viene usata la proprieta’ owl:DisjointWith tra tutte le classi di Malware in modo che si possa indicare un Malware ibrido.



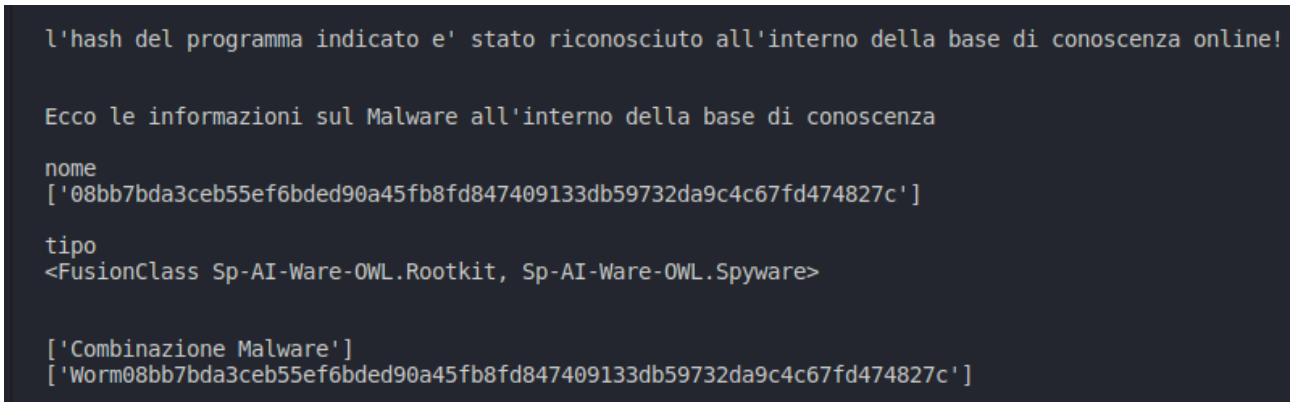
Una volta che il Malware viene classificato, nella base di conoscenza viene caricato sottoforma di istanza il valore sha del Malware e la sua categoria.

A questo punto , all’utente viene data la possibilita’ di caricare informazioni piu’ specifiche e tecniche relative al Malware rilevato , come la vulnerabilita’ sfruttata, tipo di propagazione, tipo di attacco , Proprieta’ del Malware, Programma Ospite, Obbiettivo d’attacco, Linguaggio di programmazione usato e combinazione con altri Malware.

NB: le informazione specifiche del funzionamento sono in genere vastissime e quelle presenti sono rappresentative. Manipolando

l’ontologia, e’ possibile in futuro inserire tutte le informazioni di cui si dispone.

Finito questo procedimento , il Malware verra’ segnalato ogniqualvolta verra’ analizzato un software con lo stesso sha . Esso non verra’ piu’ scansionato ma fornira’ tutti i dettagli indicati in precedenza

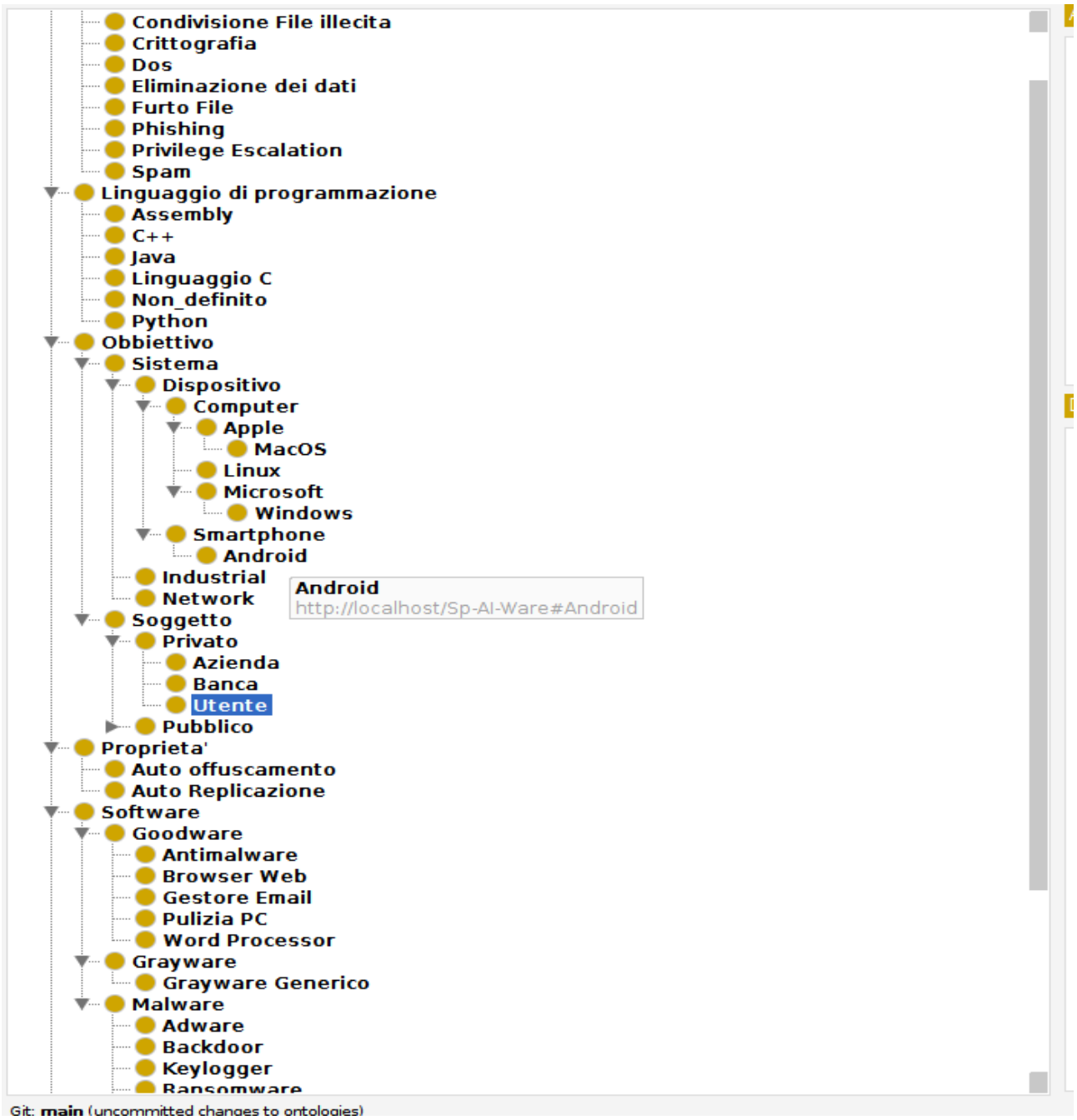


l’identificativo di ogni proprieta’ consiste nel nome della proprieta’ associato al sha per identificare un’univoca istanza della proprieta’ (ad esempio un attacco Dos fatto da un certo Malware e’ concettualmente e tecnicamente diverso da un attacco Dos fatto da un altro Malware anche se il fine e’ lo stesso)

NB: Anche i linguaggi di programmazione come tutte le altre classi sono trattati come classi istanziabili ed e’ possibile riferirsi ad esempio a 2 python diversi per lo stesso Malware. Questo e’ stato fatto con l’assunzione che ogni istanza e’ unica (ad esempio lo stesso Malware potrebbe utilizzare Python 2.16 o Python 3.7 ecc)

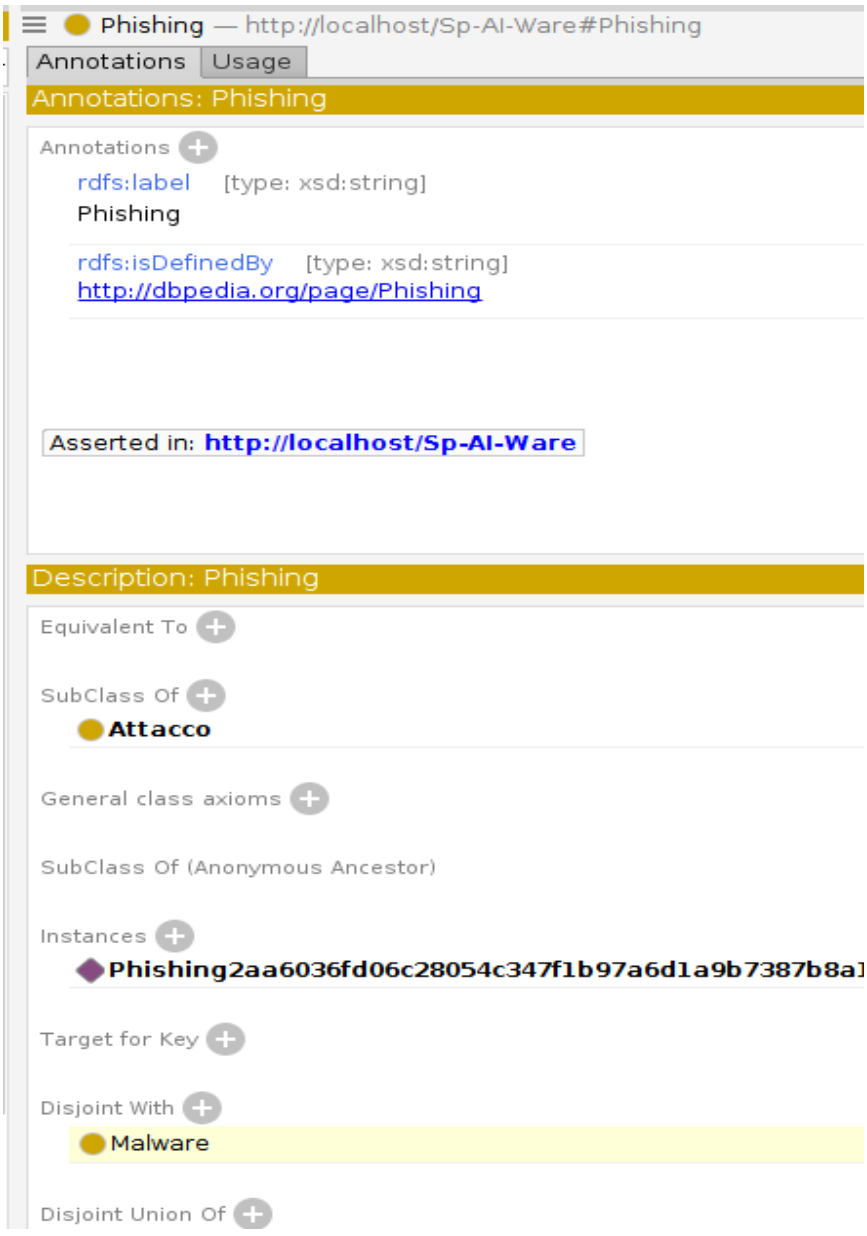
ONTOLOGIA OWL/RDF

Come detto in precedenza, le informazioni vengono caricate nel dettaglio in un ontologia OWL/RDF.
Lo scheletro dell'ontologia e' stato progettato utilizzando Protége e la sua consistenza e' stata verificata con il reasoner Hermitt. Le classi e le proprieta' create , sono progettate in modo che il Malware sia l'entita' principale che si interfaccia con tutte le altre classi



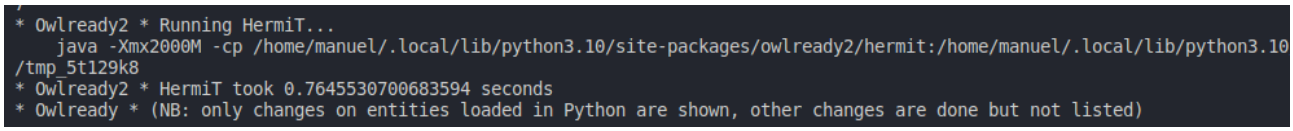
Ogni classe e’ stata fornita di una proprieta’ rdfs:label per avere una rappresentazione testuale e (se presente) di un associazione all’IRI della corrispondente risorsa su Dbpedia tramite la proprieta’ rdfs:isDefinedBy.

NB: non e’ stato sempre possibile associare una risorsa dell’ontologia a una risorsa su Dbpedia poiche’ la proprieta’ dbo:abstract utilizzata per fornire una descrizione della classe, a volte non era presente o era presente solo in lingua inglese .



Dalla propsettiva dell’interfaccia da Sp-AI-Ware, l’ontologia viene manipolata in maniera programmatica attraverso la libreria Owlready2 dove vengono inserite delle istanze del Malware

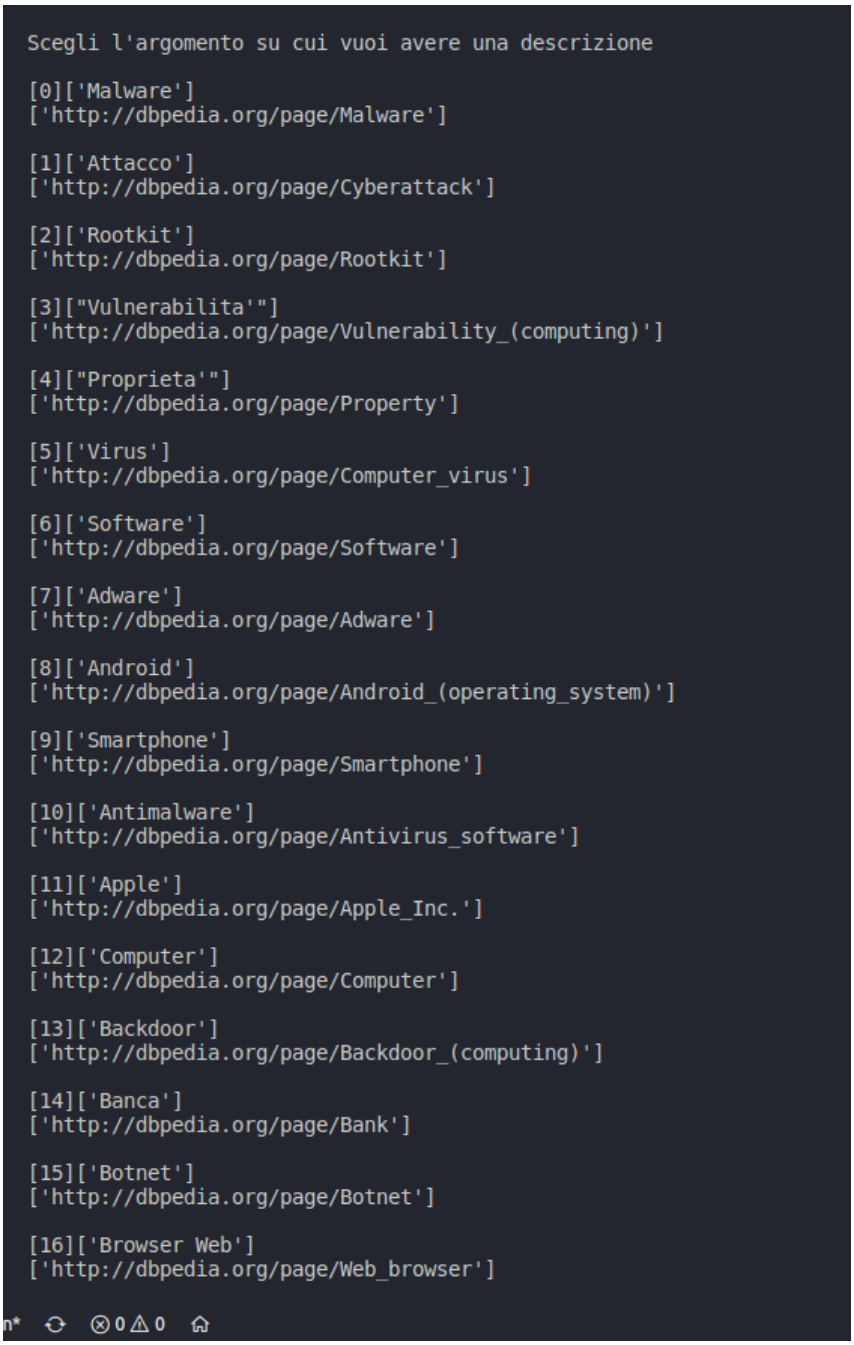
scansionato o delle altre classi e successivamente, a ogni operazione di inserimento, Hermitt verifica la consistenza dei dati inseriti e inferisce nuove proprieta’ sulle istanze a partire da quelle esistenti



NB: il file contentente l’ontologia viene salvato in locale nella cartella Ontologies. Tuttavia l’idea alla base della sua creazione e’ quella di avere informazioni condivise sul web di facile interpretazione per gli esperti del dominio della Cybersecurity e comprensibili anche dalla macchina, creando cosi una base di dati non strutturata , flessibile e di facile fruizione. Per fare questo, l’ontologia dovrebbe essere sempre disponibile online su un webserver (tipo Apache o Nginx) e dovrebbe fornire un endpoint per l’interrogazione di query Sparql da parte di altri utenti o programmi . Questo esula dallo scopo del progetto ma andrebbe utilizzata cosi per fare in modo di appartenere all’ecosistema del Semantic Web in tutto e per tutto.

DBPEDIA E SPARQL

Nel menu' principale di Sp-AI-Ware oltre alla scansione dei vari eseguibili, e' possibile usufruire di un golossario delle varie classi all'interno dell'ontologia . Questo avviene attraverso un'interrogazione all'endpoint Sparql presente su Dbpedia.org e in particolare , Sp-Ai-Ware effettua una query per ottenere il valore della proprieta dbo:abstract che rappresenta la descrizione della classe indicata . La API utilizzata come ponte tra Python e l'endpoint Sparql di Dbpedia e' SparqlWrapper.



NB: Dbpedia e le sue risorse sono in costante cambiamento: informazioni che mancano potrebbero essere inserite da un momento all’altro e informazioni presenti potrebbero essere eliminate o modificate. Al momento degli ultimi test effettuati su Sp-AI-Ware, tutte le query hanno funzionato correttamente .

SCELTE PROGETTUALI

Aldila’ delle varie librerie utilizzate per svolgere le varie funzioni di Sp-AI-Ware, le due scelte progettuali piu’ importanti sono state sul tipo di apprendimento del dataset e della rappresentazione della base di conoscenza .

***Apprendimento*:** anche se il dataset fornisce la variabile target is_malicious, si potrebbe pensare di optare per delle tecniche di apprendimento non supervisionato , cercando di effettuare un clustering delle varie osservazioni nel dataset . Tuttavia, come detto in precedenza , il funzionamento di un Malware non e’ assolutamente di facile comprensione , poiche’ vengono create nuove tipologie e versioni ogni giorno a un ritmo disarmante . Pertanto la scelta finale e’ ricaduta su apprendimento supervisionato con classificazione binaria.
La scelta dei modelli e in particolare del modello a Stack e’ giustificata da dati nel dataset che non sono assolutamente di facile interpretazione . Supponendo di cambiare dominio, sarebbe facile evidenziare relazioni tra features e target. Ad esempio si potrebbe mostrare una significativa relazione tra peso e rischio di infarto di una persona, traendo conclusioni sui dati e sul modello da usare. In questo caso , la natura dei dati e’ estremamente tecnica e a volte ambigua, pertanto il modello a Stack dei 4 modelli (SVM, Regressione Logistica, Alberi di Decisione e Random Forest) cerca di ottenere il meglio delle predizioni di modelli che generalmente performano bene su classificazioni di tipo binario.

Base di conoscenza: nel campo della Cybersecurity , la condivisione delle informazioni deve essere necessariamente tempestiva e le informazioni condivise devono essere fruibili agli esperti di dominio ma anche a software Antivirus e simili. Con la continua evoluzione del Web 3.0 (Semantic Web) ma anche degli stessi Malware, una base di conoscenza reputata ideale e' proprio una ontologia nel mondo della Cybersecurity integrata con programmi Antimalware come Sp-AI-Ware.

CONCLUSIONI

Sp-AI-Ware e’ un progetto universitario che non puo’ avere la pretesa di spacciarsi per software di Sicurezza Informatica professionale. Le competenze necessarie per implementare un software simile , esulano dal dominio di conoscenza di uno studente. Software di questo tipo vengono sviluppati e mantenuti da ben piu’ di una persona . Tuttavia Sp-AI-Ware vuole fornire un esempio della potenza di tecnologie riguardanti l’intelligenza artificiale e dell’utilizzo di strumenti quali il Web Semantico che rappresentano il futuro dell’informatica. Questo progetto potra’ essere ampliato e potenziato magari fornendo la possibilita’ di ampliare le classi stesse dell’ontologia man mano che nuovi Malware vengono scoperti e identificati o semplicemente potra’ essere uno spunto per programmi Antimalware piu’ sofisticati dei piu’ classici .

ARGOMENTI TRATTATI E RIFERIMENTI BIBLIOGRAFICI

Apprendimento Supervisionato:
https://artint.info/2e/html/ArtInt2e.Ch7.html

Sistemi Basati su conoscenza e Ontologie:
https://artint.info/2e/html/ArtInt2e.Ch14.html

Blue Hexagon Open Dataset for Malware Analisys:
https://whyisyoung.github.io/BODMAS/

Web Semantico W3 Consortium
https://www.w3.org/standards/semanticweb/